PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-265781
(43)Date of publication of application : 15.10.1993

(51)Int.Cl. G06F 9/46

(21)Application number : 04-325628
(22)Date of filing : 04.12.1992

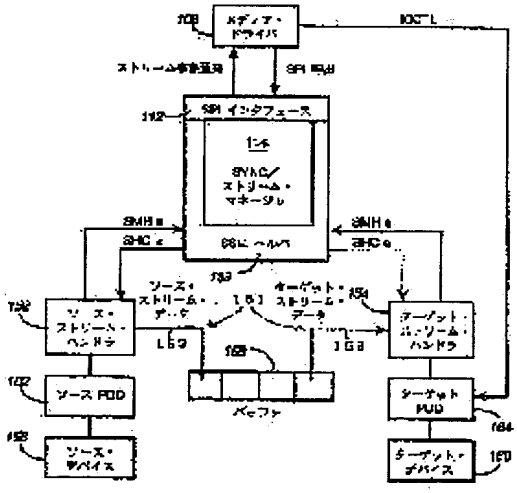
(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : DINALLO CHRIS A
KOVAL MICHAEL J
LAWTON WILLIAM W
PAULAT JR MARTIN J
TYLER JOHN G
WINTERS SCOTT L
ALLRAN GARY G

(30)Priority
Priority number : 91 816517 Priority date : 31.12.1991 Priority country : US

(54) MULTIMEDIA DATA PROCESSING SYSTEM

(57)Abstract:
PURPOSE: To provide a multimedia system which can transmit a large amount of data continuously in real time.

CONSTITUTION: This system includes a multimedia application program and a multitasking operating system. The extension of the operating system controls data streaming from a source device 158 to a target device 160 to provide real-time continuous streaming. The extension provides central buffer management by a user buffer option, the support of 2-level priority for a data stream handler, the support of an interleaved stream, and the event detection and information of a data stream.



LEGAL STATUS

[Date of request for examination] 04.12.1992
[Date of sending the examiner's decision of rejection] 01.08.1995
[Kind of final disposal of application other than the examiner's decision of rejection or

application converted registration]	
[Date of final disposal for application]	
[Patent number]	2521016
[Date of registration]	17.05.1996
[Number of appeal against examiner's decision of rejection]	07-23828
[Date of requesting appeal against examiner's decision of rejection]	30.10.1995
[Date of extinction of right]	17.05.1999

(19)日本国特許庁（ＪＰ）　(12)特　許　公　報（Ｂ２）　(11)特許番号

第2521016号

(45)発行日　平成８年(1996)７月31日　(24)登録日　平成８年(1996)５月17日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	9/46	３４０		Ｇ０６Ｆ	9/46	３４０Ａ
	13/38	３４０	9188－5Ｅ		13/38	３４０Ｃ

請求項の数３(全 38 頁)

(21)出願番号　特願平4－325628

(22)出願日　平成４年(1992)12月４日

(65)公開番号　特開平5－265781

(43)公開日　平成５年(1993)10月15日

(31)優先権主張番号　８１６５１７

(32)優先日　1991年12月31日

(33)優先権主張国　米国（ＵＳ）

前置審査

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　クリス・エイ・ディナロ
アメリカ合衆国33431、フロリダ州ボカ・ラトン、ハーバー・ドライブ　640

(74)代理人　弁理士　坂口　博　（外２名）

審査官　吉岡　浩

最終頁に続く

(54)【発明の名称】　マルチメディア・データ処理システム

(57)【特許請求の範囲】

【請求項１】プロセッサと、

複数のバッファを有し、少なくとも１つのマルチメディア・アプリケーション・プログラム及びマルチタスキング・オペレーティング・システムを記憶するメモリと、

メディア・データを供給するソース・デバイスと、

前記メディア・データを受け取るターゲット・デバイスと、

前記オペレーティング・システムの下に動作し、前記ソース・デバイスから前記ターゲット・デバイスへデータをストリーム転送するデータ・ストリーミング手段とを備え、

前記データ・ストリーミング手段が、

前記アプリケーション・プログラム中のストリーム生成命令の実行に応答して、前記ソース・デバイスから前記バッファへデータをストリーム転送するためのソース・スレッドを生成し、前記アプリケーション・プログラム中の開始命令に応答して、前記ソース・スレッドを実行することにより前記ソース・デバイスから前記バッファへデータを読み込むソース・ストリーム・ハンドラと、

前記ストリーム生成命令の実行に応答して、前記バッファから前記ターゲット・デバイスへデータをストリーム転送するためのターゲット・スレッドを生成し、前記開始命令に応答して前記ターゲット・スレッドを実行することにより前記バッファから前記ターゲット・デバイスへデータを読み出すターゲット・ストリーム・ハンドラと、

前記ソース・デバイスからのデータの終わりに達するまで前記ソース・スレッド及び前記ターゲット・スレッドを交互に実行させることによって前記ストリーム転送を

実時間で連続的に行うためのストリーム・マネージャとを含み、
前記ソース・スレッド及び前記ターゲット・スレッドは前記オペレーティング・システムの下で選択的に実行されるディスパッチ可能な実行単位であることを特徴とする、
マルチメディア・データ処理システム。
【請求項２】前記プロセッサはオペレーションの異なる優先レベルを有し、各前記ストリーム・ハンドラは、ある優先レベルにおいて実行されるダイナミック・リンク・ライブラリ（ＤＬＬ）・ストリーム・ハンドラと、前記ある優先レベルよりも高い優先レベルにおいて実行される物理デバイス・ドライバ（ＰＤＤ）・ストリーム・ハンドラを含むこと、
を特徴とする請求項１記載のマルチメディア・データ処理システム。
【請求項３】前記ソース・デバイスからの前記データは順次的にインタリーブされた複数のレコードに含まれ、前記ソース・ハンドラは各バッファを前記レコードにより充填すること、
を特徴とする請求項１又は２記載のマルチメディア・データ処理システム。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明はデータ処理分野に関し、特に、マルチタスキング・オペレーティング・システムの制御の下で、データ・ストリーム転送を連続的に、実時間方式で発生するパーソナル・コンピュータを基本とするマルチメディア・システムに関する。
【０００２】
【従来の技術】マルチメディア・システムはテキスト、グラフィックス、ビデオ、イメージ、アニメーション、オーディオなどの種々の組合せにおいて、様々なマルチメディア材料を提供する。こうしたシステムはハードウェアとソフトウェアの組合せにより構成される。ハードウェアには種々のマルチメディア装置が接続されるパーソナル・コンピュータが含まれる。ハードウェアはオペレーティング・システム及びマルチメディア・アプリケーション・プログラムの制御の下で機能する。
【０００３】マルチメディア・アプリケーションは装置間、或いはシステム・メモリから装置へ、或いはその反対方向へ、連続的に実時間で大量のデータを転送するために、オペレーティング・システムに重い負荷を課す。マルチメディア・システムはこれらの大きなデータ・オブジェクトを転送するための柔軟でしかも一貫した手段を支持し、またこの活動を実時間で正確に制御しなければならない。新たなマルチメディア装置及びデータ・タイプを追加するためには、最小限度の新たなシステム拡張コードが必要となる。実行時に必要な実メモリのトータルは、システム性能が低下しないように最小化されねばならない。また、複雑なデータ・タイプ及びインタリーブ式のデータ・オブジェクトを操作する装置の支持も必要である。さらに、システムの最適な特権レベルにおいて、各マルチメディア・データ転送制御手段が実現されなければならない。
【０００４】マルチメディア・データ・タイプ及び装置を支持するオペレーティング・システムの拡張は、多数の異なるマルチメディア入出力装置を制御し、また実時間の制限範囲において、大量のマルチメディア・データ・オブジェクトを転送或いはストリームする機能を提供することが必要がある。
【０００５】マルチメディア・アプリケーションは、ビットマップの連続的な表示、デジタル化オーディオ波形或いはＭＩＤＩオーディオ・シーケンス、或いはアナログ・マイクロフォンまたはライン源からのデジタル化オーディオ入力などの大量のデータの入出力を制御する。アプリケーションはこれら全てのデータ・フローを実時間クロックにより制御する。すなわち、プログラムにより制御される特定の事象は、時間的に明確に定義された時点において発生されなければならず、これらの時点は非常に正確に（例えばミリ秒単位で）定義される。
【０００６】ＯＳ／２制御プログラム・サービス（ＤＯＳ呼出、或いはＡＩＸまたはＵＮＩＸなどの他のオペレーティング・システムにおける類似のサービス）だけを考慮すると、アプリケーション・プログラム・インタフェースにおいてこのレベルの機能を制御するには、非常に複雑で装置依存性の高いデータ転送制御モジュールを要する。こうしたモジュールが生成されたとしても、各入出力オペレーションを制御するスレッド（thread）がその要求時間内に実行される保証はない。この問題を指摘すると、アプリケーションは複雑なセマフォー（semaphore）論理を追加し、入出力制御スレッドを時間的に厳密なもの、すなわちタイム・クリティカル・スレッドにする必要が生じる。マルチメディア・アプリケーションでは一般にデータ・スループットが高負荷であるから、マルチタスキング・オペレーティング・システムのもとでは、しばしば、割当時間間隔内にデータを宛先装置に転送することが阻害される。これらの実時間要求を満たせないと、結果的にマルチメディア表現における視覚的或いは聴覚的な欠陥を発生させる。
【０００７】
【発明が解決しようとする課題】実時間における連続的な大量のデータの転送要求は、一般化された転送機構が解決しなければならない一連の相互に依存する問題を提示する。これらの問題を以下に示す。
【０００８】１．非常に負荷の大きいデータ入出力要求を有するスループット重視のアプリケーション
解像度が６４０ｘ４８０ｘ１６の非圧縮動画ビデオでは、おおよそ３Ｍｂ／秒を必要とする。例え高度に圧縮されたとしても、デジタル動画ビデオ・データ・ストリ

ームは６０Ｋｂ／秒を必要とする。８或いは１６ビットのデジタル・オーディオ波形のデータ転送をシナリオに追加すると、スループットの負荷は８０Ｋｂ／秒或いはそれ以上に増加する。このスループット負荷を連続的に支持する一方で、マルチタスキング・システムがファイル入出力及びキーボード／マウス装置の制御などの他のアプリケーション機能を実行することが、一般化されたデータ・ストリーミング機構において最も重要な要求である。しかしながら、ユーザ・インタフェース（聴覚的及び視覚的）におけるマルチメディア・データ表示は実時間依存であるため、これらのデータ転送は非常にタイトな実時間制限において発生しなければならない。

【０００９】２．複数の異なるハードウェア入出力装置の制御

マルチメディア・アプリケーションは典型的には種々の入出力装置を制御し、ユーザ・インタフェースのオーディオ及びビデオ分野における表現を管理する。波形獲得及び再生を支援するオーディオ装置、デジタル・オーディオ・コンパクト・ディスク再生、楽器デジタル・インタフェース（ＭＩＤＩ）入出力、及び音声発生などが含まれる。同様に、ＮＴＳＣ（或いはＰＡＬ）ビデオのデジタル化、圧縮／伸長、及び獲得を支援するビデオ装置が、ルーチン的に、マルチメディア提示及び／或いはシナリオに含まれる。一般化されたデータ・ストリーミング機構は、マルチメディア・システムの拡張に関する大きな変更を要求することなく、これらの及びその他のマルチメディア装置を柔軟に支持する必要がある。

【００１０】３．装置及びデータ・タイプ間における一貫した制御サービスの提供

データ・ストリーム機構は、マルチメディアの創作及び表現の過程に含まれる異なる装置及びデータ・タイプ間で、一貫したサービスによって制御されねばならない。マルチメディア・データ標準が発展する過程において新たなデータ・タイプが導入される時、この一般化されたストリーミング機構は実質的に影響されてはならない。制御サービスは新たなデータ・タイプを現存のアプリケーションに容易に組み込むことが必要とされる。同様に、これらのサービスは、現存のアプリケーションに影響を及ぼすことなく、或いはマルチメディア拡張の大きな変更を必要とせずに、新たな入出力装置のタイプ及び機能を途切れることなく追加する配慮が必要である。

【００１１】４．複数のシステム特権レベルにおける同一制御サービスの提供

マルチメディア・データ・オブジェクトは、ハードウェア入出力装置或いはシステム（またはアプリケーション）・メモリにおいて開始（或いは終了）される。データ・ストリーミングがハードウェア装置と共に開始或いは終了すると、伝送機構はハードウェア装置の振舞いを直接的に制御し、これにより装置依存型となるのではなく、間接的に制御できなければならない。しかしながら、最近のオペレーティング・システムにおいて共通な特徴である、複数の特権レベルを支持する性質においては、最適性能を出すために、伝送機構（〝ストリーム・ハンドラ〟）が装置固有の物理デバイス・ドライバに最高のシステム特権レベルで接続されることが必要とされる。これはハードウェア装置ドライバが通常実行されるレベルである。このように性能的な理由により、デバイス・ドライバ・レベルのストリーム・ハンドラを作成することが望ましいが、ストリーム・ハンドラ・デバイス・ドライバはそれ自身ハードウェア依存であってはならない。全てのハードウェア特有のコードは、デバイス固有の物理デバイス・ドライバに含まれる必要がある。あるいは、データ・ストリームの出所または宛先として物理的装置が含まれない場合には、ストリーム・ハンドラはより低い特権レベルで実行され、しかも最適性能を達成可能である。このようにデータ伝送機構は複数の特権層に渡り分散されている必要があるため、同一のストリーム制御サービスが、ストリーム・ハンドラが実行される所与のレベルにおいて提供されねばならない。異なる特権レベルにおける複数の入力ポイントが、冗長コードの組込み無しに提供されることが必要である。

【００１２】５．最小限の影響により新たな装置／データ・タイプへと拡張する機能

新たなマルチメディア・データ標準及び装置タイプは、絶えず作成される。この状況は将来的にも継続するものと思われ、データ伝送機構は新たな装置及びデータ・タイプを支持するように、容易に且つ不断に適応することが必要である。

【００１３】６．ストリーミング中における物理メモリに対する要求を最小限に維持する

マルチメディア・アプリケーションにより課せられる重いスループット負荷にも関わらず、データ・ストリーミング中は常にメモリ・リソースの保存が必要である。アプリケーションが物理的なメモリ・マネージャになる必要のないように、バッファ管理サービスが提供されねばならない。しかしながら、いくつかのアプリケーションはマルチメディア・データ・オブジェクトをメモリ内に作成するため、これらのアプリケーションについてはメモリ・バッファの直接的な制御を有する必要はない。この機能はデータ・ストリーミング機構及びその制御サービスにより、不断に支持されねばならない。

【００１４】７．ストリーミング中のＣＰＵローディングを最小限に維持する

メモリが慎重に管理されねばならない逼迫したリソースであるように、ＣＰＵサイクルは別の重要なリソースであり、マルチメディア・アプリケーションの実行中に浪費されてはならない。データ・ストリーミング機構はハードウェア間のデータ伝送機能（例えばバス・マスタ・データ転送）を利用して、ＣＰＵ負荷を最小に維持する。更に、データ転送機構にデバイス・ポーリング技術

を使用して装置を制御するのではなく、可能な時に割込みにより駆動されるデバイス入出力が利用され、ＣＰＵの負荷を軽減する。

【００１５】以下の定義は本発明を説明するフレームワークを構成するために提供される。

【００１６】データ・ストリーム・・・データ・チャネルを介し、ソース・デバイス（或いはメモリ・アドレス）からターゲット・デバイス（或いはメモリ・アドレス）へ連続的にデータ転送するためのソフトウェア手段。

【００１７】ダイナミック・リンク・ライブラリ（ＤＬＬ）・・・ＯＳ／２アーキテクチャでは、１６ビット或いは３２ビットの実行可能モジュールが特権レベル２または３で実行され、別の実行可能モジュール（例えば．ＥＸＥ）の外部参照を解決するためにメモリにロードされる。ＤＬＬはタスク・コンテキストにおいてのみ実行可能である。

【００１８】物理デバイス・ドライバ（ＰＤＤ）・・・ＯＳ／２アーキテクチャでは、１６ビットの実行可能モジュールが特権レベル０（リング０）で実行され、典型的には入出力装置のオペレーションを制御する。ＰＤＤはタスク・コンテキスト或いはハードウェア割込みコンテキストにおいて実行される。

【００１９】ストリーム・・・データをソースから宛先（或いはターゲット）に連続的に転送する。

【００２０】

【課題を解決するための手段】本発明は一般化されたデータ・ストリーム転送を改善し、様々な入出力装置間のマルチメディア・データ伝送機構、アプリケーション・メモリ、及び上述の問題を解決し更に上述の要求を満足するように設計されたシステム・メモリを提供する。伝送機構は複数の実行特権レベルを組込むマルチタスキング・オペレーティング・システムを拡張することによって実施される。この機構はデータ・ストリーミングと称される連続的な、実時間のデータ転送をサポートする。

【００２１】

【実施例】図を参照しながら説明すると、図１は一般的なデータ処理システムを示し、例えばＯＳ／２バージョン２．０などのマルチタスキング・オペレーティング・システム下で動作するパーソナル・コンピュータ１０を含み、アプリケーション・プログラムを実行する。コンピュータ１０はマイクロプロセッサ１２を含み、これはローカル・バス１４に接続され、ローカル・バス１４はバス・インタフェース制御装置（ＢＩＣ）１６、数理コプロセッサ１８、及びＳＣＳＩ（smallcomputer system interface）アダプタ２０に接続される。マイクロプロセッサ１２は好適には、８０３８６或いは８０４８６マイクロプロセッサなどの８０ｘｘｘマイクロプロセッサ・ファミリの１つであり、ローカル・バス１４はこうしたプロセッサのアーキテクチャに準じる従来のデータ、アドレス及び制御ラインを含む。アダプタ２０もまたＳＣＳＩバス２２に接続され、ＳＣＳＩバス２２はＣドライブとして指定されるＳＣＳＩハード・ドライブ（ＨＤ）２４に接続される。バス２２は他のＳＣＳＩ装置（図示せず）にも接続可能である。アダプタ２０はまたＮＶＲＡＭ３０及び読出し専用メモリ（ＲＯＭ）３２にも接続される。

【００２２】ＢＩＣ１６は２つの主な機能を実行する。１つはメモリ制御装置が主メモリ３６及びＲＯＭ３８をアクセスする機能である。主メモリ１６はダイナミック・ランダム・アクセス・メモリ（ＤＲＡＭ）であり、マイクロプロセッサ１２による実行のためのデータ及びプログラムを格納する。ＲＯＭ３８はＰＯＳＴプログラム４０及びＢＩＯＳ４２を格納する。ＰＯＳＴプログラム４０はコンピュータ１０がパワー・オン或いはキーボード・リセットによりスタートされる時に、標準のパワー・オン及びシステムのセルフ・テストを実行する。アドレス及び制御バス３７はＢＩＣ１６をメモリ３６及びＲＯＭ３８に接続する。データ・バス３９はメモリ３６及びＲＯＭ３８をデータ・バッファ４１に接続し、データ・バッファ４１は更にバス１４のデータ・バス１４Ｄに接続される。制御ライン４５はＢＩＣ１６及びデータ・バッファ４１を相互接続する。

【００２３】ＢＩＣ１６の他の主な機能は、バス１４とマイクロ・チャネル（ＭＣ）・アーキテクチャに準じて設計される入出力バス４４間のインタフェースである。バス４４は更に入出力制御装置（ＩＯＣ）４６、ビデオ信号プロセッサ（ＶＳＰ）４８、デジタル信号プロセッサ（ＤＳＰ）４９、及び複数の拡張コネクタ（ＥＣ）或いはスロット５０に接続される。ＶＳＰ４８は更にビデオＲＡＭ（ＶＲＡＭ）６０及びマルチプレクサ（ＭＵＸ）６２に接続される。ＶＲＡＭ６０はモニタ６８の画面上の表示を制御するテキスト及びグラフィック情報を格納する。ＭＵＸ６２は更にデジタルーアナログ変換器（ＤＡＣ）６６、及び、ビデオ機能バス（ＶＦＢ）に接続可能なコネクタ或いはターミナル７０に接続される。ＤＡＣ６６は従来の出力画面或いは表示を提供するモニタ６８に接続され、これはユーザによって観察される。

【００２４】ＩＯＣ４６は複数の入出力装置のオペレーションを制御し、これらの装置にはＡドライブとして指定されるフロッピ・ディスク・ドライブ７２、プリンタ７４、及びキーボード７６が含まれる。ドライブ７２は制御装置（図示せず）及び取り外し可能フロッピ・ディスク或いはディスケット７３を含む。ＩＯＣ４６はまたマウス・コネクタ７８、シリアル・ポート・コネクタ８０、及びスピーカ・コネクタ８２に接続され、これらのコネクタにより種々のオプション装置がシステムに接続される。

【００２５】ＤＳＰ４９は更に命令ＲＡＭ８４、データＲＡＭ８６、アナログ・インタフェース制御装置（ＡＩ

Ｃ）８８、及びオーディオ制御装置９０に接続される。ＲＡＭ８４及び８６はそれぞれＤＳＰ４９が信号処理するために使用する命令及びデータを保持する。オーディオ制御装置９０は種々のオーディオ入出力を制御し、また複数のコネクタ９２に接続される。これらのコネクタにより様々な装置がシステムに接続可能となる。こうした装置にはヘッドフォン、マイクロフォン、スピーカ、楽器デジタル化インタフェース（ＭＩＤＩ）、及びオーディオ・ライン・イン及びライン・アウト機能を必要とする装置が含まれる。他の様々なマルチメディア装置（MMD）９６も、ＥＣ５０及びアダプタ・カード９４を介してシステムに接続可能である。

【００２６】メモリ３６はシステムで実行される様々なプログラムを格納し、これらのプログラムにはマルチメディア・アプリケーション・プログラム（MMAP）１０２を含むアプリケーション・プログラム１００、及びオペレーティング・システム９８が含まれ、後者はsync/stream sub-system（Ｓ／ＳＳ）１０４を含む拡張を有する。ここで図１は典型的なマルチメディア・システムを表すが、オペレーティング・システムは汎用目的であり、図１で示される構成とは異なる構成を有するデータ処理システムについても実行或いは制御できるように設計されることを述べておく。本発明は主にＳ／ＳＳ１０４及びそのオペレーティング・システム１００との対話において実施され、これについて次に説明することにする。

【００２７】SYNC/STREAM SUB-SYSTEM

【００２８】図２を参照すると、マルチメディア・アプリケーション・プログラム１０２はオペレーティング・システム９８の上の層で実行され、マルチメディア制御インタフェース（ＭＣＩ）を介して、マルチメディア環境における装置を制御するためのＭＣＩコマンドを送信することにより通信する。いくつかの基本コマンドにはpause、play、record、resume、seek、save、set、stopなどがある。こうしたコマンドはオペレーティング・システム９８によりメディア・デバイス・マネージャ（MDM）１０６に転送される。MMAP用のアプリケーションプログラミング・モデルはOS/2 Presentation Manager プログラミング・モデルの論理拡張であり、オブジェクト指向のメッセージング構成体及び手順（call及びreturn）プログラミング・インタフェースの両方を組み込む。ＭＣＩはアプリケーション開発者、及び、ビデオ及びオーディオ家庭演奏システムの開発者などのユーザにビューを提供する。オペレーションはメディア装置として知られるメディア情報用のプロセッサを制御することにより実行される。メディア装置は内部或いは外部ハードウェア装置であるか、或いは、低レベルのハードウェア要素及びシステム・ソフトウェア機能を操作することにより、定義された一連のオペレーションを実行するソフトウェア・ライブラリが存在する。複数のメディア装置をシナリオに含むことが可能であり、これらの装置は再生を同期させるために、グループとして割り付けられて制御される。

【００２９】マルチメディア・アプリケーションはシステムの実時間の振舞いにおける２つの面を制御する必要があり、ある装置から別の装置への大量のデータの転送、及び関連する事象の同期がそれに相当する。プログラムにより制御される事象はユーザの直接の制御の下で認識されることが必要であり、基本となるシステム機能はこれらの事象が予測可能であり実時間において発生することを容易にし、また保証する。マルチメディア・アプリケーションの制作者は実時間クロックを基本に動作するプログラムを作成するべきである。

【００３０】ＭＣＩは２つのレベルの動的なリンク・ライブラリ（ＤＬＬ）を有し、これらはMDM１０６及びメディア・ドライバを含み、後者はオーディオ・メディア・ドライバ１０８及びＭＩＤＩメディア・ドライバ１１０を含む。MDM１０６はメディア装置のリソース管理を提供する。また、MDM１０６はメディア装置へのアクセスの競合を解決し、アプリケーション開発者にハードウェアに独立なリソースの考え方を提供する。メディア・ドライバは動的なリンク・ライブラリであり、メディア装置の機能を実施する。メディア・ドライバはハードウェア装置のサービス或いはソフトウェアを呼出し、それらの機能を実施する。メディア・ドライバは直接的にはハードウェア装置を制御しない。その代わり、コマンドをストリーム・プログラミング・インタフェース（ＳＰＩ）１１２及びsync/stream マネージャ（ＳＳＭ）１１４を介して、Ｓ／ＳＳ１０４に転送する。ＳＳＭ１１４は同期及びストリーミング活動を制御する。ＳＳＭ１１４は２つの主な機能を実行し、その第１は連続的な実時間のデータ・ストリーミングを提供するためのデータ・ストリーミングの制御であり、これは本出願の目的でもある。第２の機能はデータ・ストリームの同期を含み、こうした機能の詳細は上述の関連するアプリケーションにより実施される。

【００３１】ストリーム・ハンドラがシステム・カーネル・レベル及びアプリケーション・レベルの両方で必要とされる。いくつかのデータ・ストリームは、ストリーム・ハンドラとリング０の特権レベルを有する物理デバイス・ドライバとの間の直接接続により最適に制御される。こうしたストリーム・ハンドラはOS/2Interdevice Driver Communication（ＩＤＣ）に基づく共通のインタフェースを使用してＰＤＤと通信する。他のデータ・ストリームは特定の物理装置にマップ化可能なデータ・ソース或いはターゲットとは関連せず、ＤＬＬによりリング３の優先レベルで制御される。点線１１３は一般にどの項目が異なる優先レベルにおいて動作するかを示している。ＳＳＭ１１４において、扱っているタスクに応じて、いくつかのルーチンはあるレベルで動作し、他のル

ーチンは別のレベルで動作する。

【００３２】各ストリーム・ハンドラはプログラマブルであり、ストリーム・プロトコルに準じてストリーミング可能である。ＳＳＭ１１４の観点より、全てのストリーム・ハンドラは同様な責務を負う。各ハンドラは１つ以上のデータ・ストリームに対応するソース或いはターゲットに相当するように設計され、各データ・ストリームはデータを独立に転送する。マネージャ１１４はストリーム・ソースを適切なストリーム・ターゲットに接続してデータ・フローを維持し、またストリームの終了に際し、種々のリソースをクリーニングする役割を担う。更に、ストリーム・ハンドラはデバイス依存型のモジュールではない。各ストリーム・ハンドラは特定の事前に定義されたタイプのデータのストリーミングを支持するが、データはあるハンドラから次のハンドラに、ハードウェアの振舞いに関する知識無しに転送される。また、オーディオ・ストリーム・ハンドラ１３４は互換性を有するオーディオ装置のＰＤＤと、完全にハードウェア独立形式で通信する。互換であるためには、ＰＤＤは標準オーディオ・デバイス・ドライバ・インタフェースＩＯＣＴＬの場合同様、ＩＤＣインタフェースに準じることが必要である。図示されるように、ストリーム・ハンドラ１１４は複数のストリーム・ハンドラ動的リンク・ライブラリ（ＤＬＬ）１１６ー１２６と対話する。これらはそれぞれＭＩＤＩマッパー、ＣＤ／ＤＡ、メモリ、分割ストリーム、ＣＤ－ＲＯＭ／ＸＡ、及びファイル・システムのストリーム・ハンドラに対応する。ファイル・システム・ストリーム・ハンドラＤＬＬ１２６はファイル・システム１３０をアクセスするために、マルチメディア入出力（ＭＭＩＯ）マネージャ１２８を呼出す。

【００３３】ストリーム・マネージャ１１４はまた、ストリーム・マネージャ・ヘルパ１３３を介して、オーディオ・ストリーム・ハンドラ物理デバイス・ドライバ（ＰＤＤ）１３４と対話する。１３４はストリーム・ハンドラ／ＰＤＤインタフェース１３６及び複数のＰＤＤ１３８ー１４４を介して、選択的に物理装置をアクセスする。ストリーム・マネージャ１１４はまた、デバイス間通信（ＩＤＣ）インタフェース１４６を介してインタフェース１３６と対話できる。

【００３４】図３はデータ・ストリーミング・オペレーションの一般化モデルであり、その詳細は以降の図に示されるフローチャート及びデータ構造に関連して説明される。図３は一般に単一のデータ・ストリーム１５１を示し、データがストリーム・マネージャ１１４、及びソース及びターゲット・ストリーム・ハンドラ１５２及び１５４の制御により伝送される様子を示す。複数のストリーム・バッファ１５６がメモリに割当てられ、ストリーミングに使用される。バッファ１５６はソース・デバイス１５８からのストリーム・データで充填され、ストリーム・データはターゲット・デバイス１６０にデータを転送することにより空状態となる。データ・ストリーム１５１はソース・ストリーム・データ１５３とターゲット・ストリーム・データ１５５の２つの部分から構成される。ソース・ストリーム・データのデータ経路は、ソース１５８からソースＰＤＤ１６２及びストリーム・ハンドラ１５２を介し、バッファ１５６に至る。ターゲット・ストリーム・データ１５５のデータ経路はバッファ１５６からターゲット・ストリーム・ハンドラ１５４及びターゲットＰＤＤ１６４を介し、ターゲット・デバイス１６０に至る。ソース・ストリーム・ハンドラ１５２はソースＰＤＤ１６２をアクチュエートし、ＰＤＤ１６２はソース・デバイスのオペレーションを制御する。ターゲット・ストリーム・ハンドラ１５４はターゲットＰＤＤ１６４をアクチュエートし、ＰＤＤ１６４はターゲット・デバイス１６０を制御する。ソース・ストリーム・ハンドラ１５２の一般的な目的は、ターゲット・デバイスがスタートする以前にストリーム・バッファ１５６の少なくとも２つを充填し、ターゲット・デバイスが一度スタートすると、全てのソース・データが転送されるまで、バッファを充填することにより、ターゲット・デバイスよりも先行することである。バッファが充填されると、ターゲット・ストリーム・ハンドラはそこからターゲット・データを獲得し、それをターゲット・デバイスに転送できる。

【００３５】メディア・ドライバ１０８はＳＰＩファンクション或いは呼出を送信し、ストリーム事象通知を受信することにより、ＳＰＩインタフェースと対話する。マネージャ１１４はＳＰＩ呼出を解釈し、それに応答して目的の機能を実行する。これはシステム・ヘルパ・コマンド（ＳＨＣ）をハンドラへ送信し、ストリーム・マネージャ・ヘルパ（ＳＭＨ）呼出をハンドラから受信することにより、ストリーム・ハンドラと対話して実行される。メディア・ドライバ１０８はまた、標準ＩＯＣＴＬコマンドにより定義される制御ファンクションを発行することにより、ＰＤＤ１６４を直接制御できる。本発明に関連する主なＳＰＩ呼出はSpiCreateStream及びSpiStartStream であり、それぞれ目的のストリームをセットアップし、続いてデータのストリーミングを開始する。これについては後に図２０及び２１を参照して更に詳細に説明する。同期されなければならない複数のストリームが存在する場合には、SpiEnableSync 呼出が実施され、これについては関連するアプリケーションにおいて詳しく説明されている。

【００３６】図４はストリーム・ハンドラ・コマンドSHC.CREATEが実行される時に、何が発生するかを表す。しかしながら、これを説明する前に、アプリケーション・プログラムに関連して、どのようにそのポイントに達するかを理解することが有益である。第１に、ＭＭＡＰはデータ・ストリームを生成するためにＭＣＩ呼出を発行する。こうした呼出はＯＳ９８により解析され、ＯＳは

メディア・デバイス・マネージャ１０６に情報を送信する。このマネージャは次に適切なメディア・ドライバを選択する。対称となるドライバはSpiCreateStream呼出をsync/streamマネージャ１１４に送信するドライバであり、sync/stream マネージャ１１４は続いてSHC.CREATEコマンドを適切なストリーム・ハンドラに発行する。

【００３７】図４はSHC.CREATEコマンドがＤＬＬソース・ストリーム・ハンドラ１５２により実行されるときの様子を示す。最初にステップ２０１で、ＯＳの下でスレッドを生成する。スレッドはＯＳのマルチタスキング機能により制御される。ステップ２０２ではスレッドに対し、実行されるタスクに適切な優先レベルを割当てる。次にスレッドはステップ２０３でセマフォー上でブロックされ、"スリープ"状態となる。優先レベルは、ＯＳがスレッドを割当て、実行するレートを制御するために使用される。不連続性が生じる場合には、優先レベルは調整される。

【００３８】ここでＯＳ／２の標準オペレーションに準じ、スレッドは個々のタスクとして扱われ、スレッドがブロックされる時、呼出が生成される時、及び復帰が生成される時などにおいて、ＯＳへの復帰を制御する。これによりオペレーティング・システムはマルチタスク方式により、システムにおける他のタスクを実行したり、ストリーミング・スレッドに復帰したりする。また、図には丸で囲まれた数値が矢印で接続されているが、これらはプログラムのフローを示している。例えば、接続印の１が図４と図２０で現れる。図２０では、接続印は呼出と復帰の両方を含むのに対し、図４の接続印ではルーチンがどこかで呼出されていることを示す呼出の矢印だけを含む。他の接続印についても同様に解釈できる。

【００３９】アプリケーション・プログラムにおいて生成されるスタート・コマンドに応答して、マネージャ１１４は SHC.STARTコマンドを最初にソース・ストリーム・ハンドラに（図４の２０６）、続いてターゲット・ストリーム・ハンドラに送信する。ソース・ストリーム・ハンドラは、ターゲット・ハンドラがストリーム・バッファに転送されるデータを使用可能になる以前に、当該バッファを充填するために最初に起動される必要がある。こうしたコマンドに応答して、ソース・ストリーム・ハンドラはステップ２０８でソース・スレッドをブロック解除する。ブロック解除或いは喚起に応答して、ソース・スレッドは次にステップ２１６で、マネージャ１１４から空（エンプティ）バッファを要求する。ステップ２１８でバッファが使用可能でないと判断されると、ステップ２２０で再びスレッドをブロックする。空バッファが使用可能な場合には、ステップ２２２でソース・デバイスからデータを読出し、バッファを充填する。次にステップ２２４で充填されたバッファをマネージャ１１４に戻す。ステップ２２６では更にバッファを充填する必要があるかを判断する。必要がある場合、ステップ２１６に分岐し、ステップ２１６からステップ２２６に至るループが形成され、このループはステップ２２６でバッファが充填される必要がないと判断される時にブレークされる。次にスレッドはブロックされる。ストリーミング・オペレーションが一度開始されると、ソース・ファイルの終わりがソース・スレッドが静止するするポイントに達するまで、バッファ充填処理が繰り返される。事象検出は図４のオペレーション・シーケンス内のアスタリスクで示されるポイントであれば挿入可能であり、例えば、キュー（cue）・ポイント検出により、転送されるデータ・レートにより決まる時間を追跡する。事象検出はまた、後に続く数多くのルーチン内でも使用される。

【００４０】図５はＤＬＬターゲット・ストリーム・ハンドラをリング３の優先レベルでオペレートする場合を示す。ターゲット・ストリーム・ハンドラはターゲット・データ・ストリームを生成するためにマネージャ１１４からSHC.CREATEを受信し、、最初にステップ２３０でターゲット・ストリーム・スレッドを生成する。次にステップ２３２でスレッドに対し優先度を割当て、ステップ２３４でセマフォー上のターゲット・スレッドをブロックする。ステップ２３６でマネージャ１１４が SHC.STARTコマンドを発行することにより、ターゲット・スレッドが喚起されると、ターゲット・スレッドはステップ２３８でブロック解除され、ステップ２４０でスレッドは喚起される。ステップ２４２ではデータ・ストリームからフルのバッファを獲得しようとする。ステップ２４４で、バッファが獲得されたかを確認し、獲得されていない場合は、ステップ２４６でターゲット・スレッドをブロックする。フルのバッファが獲得されると、ステップ２４８でバッファ内のデータをターゲット・デバイスに書込み、次にステップ２５０で空バッファをマネージャ１１４に戻す。ステップ２５２ではバッファがデータ・ストリームにおける最終バッファであるか、すなわちＥＯＳに達したかどうかが確認される。そうでない場合には、ステップ２４２に分岐が発生し、フルのバッファを獲得するための処理が繰り返される。ループはストリームの終わりが検出されるまで繰り返され、検出の際にステップ２５４でスレッドを終了する。

【００４１】図６を参照すると、ターゲット・ストリーム・ハンドラが物理デバイス・ドライバにおいてリング０の優先レベルで実施される様子が示される。ステップ２５６でSHC.CREATEコマンドが受信されると、ステップ２５８でストリーム・データ構造がセットアップされ、ステップ２６０ではターゲット・デバイスが初期化される。ステップ２６１で復帰が行われる。ステップ２６２でストリームが開始されると、ステップ２６４でフルのバッファに対する要求がマネージャ１１４に出される。ステップ２６６でバッファが提供されたかが判断される。もしそうであれば、ステップ２６８で入出力書込み

が開始され、その後ステップ２７０で復帰する。ステップ２７２によりエラーが生じていると判断され、その結果、バッファが返却されない場合には、ステップ２７４でエラーが報告され、次に復帰する。

【００４２】バッファへの書込み及びエンプティ処理により入出力オペレーションが完了すると、ステップ２７２は割込みを発生し、これはステップ２７４で割込みハンドラにより処理される。入出力オペレーションがエラー無しに達成されると、ステップ２７８は空バッファをマネージャ１１４に戻し、ステップ２６４に分岐が発生し、処理が繰り返される。入出力オペレーション中にエラーが発生すると、ステップ２８０でエラーが報告され、ステップ２８２で復帰する。ソース・ストリーム・ハンドラによるリング０の優先レベル処理は、空バッファを獲得し、フルのバッファを戻す以外は、図６の処理と同様である。

【００４３】使用可能なデータ・ファイル及びストリームには２つのタイプがあり、それらは統一或いは非インタリーブド・ストリーム、及び、分割或いはインタリーブド・ストリームである。統一データ・ストリーム及びファイルは、例えばオーディオ・データである１つのタイプのデータだけを含む。これに対し分割ストリームは、例えばオーディオ・データとビデオ・データの両方の複数タイプのデータを含む。ＣＤ－ＲＯＭ装置用に一般に生成されるファイル・タイプが例として挙げられる。図７は異なるタイプの例を示す。ファイル２８６は所与のタイプのオーディオ・データだけを含む統一ファイルを表す。こうしたファイルは複数のバッファＢ１－Ｂ３に読込まれる。バッファＢ１にはオーディオ・データの第１のセクションが充填され、バッファＢ２にはオーディオ・データの次のセクションが充填される。ファイル２８８は分割或いはインタリーブド・ファイルを表し、インタリーブされたオーディオ及びビデオ・データの複数のセクションが含まれる。セクションは、オーディオが、提示されるビデオに対応するように、対に構成される。こうしたデータはレコードの如く、関連するバッファＢ４－Ｂ６に転送される。すなわち、第１のオーディオ・セクションＡ１はＢ４においてオーディオ・レコードＡＲ１に、第１のビデオ・セクションＶ１はＢ４のビデオ・レコードＶＲ１に、また第２のオーディオ・セクションＡ２はＢ４のＡＲ２となる。次に、セクションＶ２、Ａ３及びＶ３がバッファ５にレコードＶＲ２、ＡＲ３及びＶＲ３として転送される。バッファＢ６も同様にＡＲ４及びＶＲ４により充填される。明らかなように、この例においては、バッファＢ４－Ｂ６は、各々、３つのレコードを保持するサイズとなる。

【００４４】ストリームがマネージャ１１４により生成されると、複数のバッファは各ストリームに割当てられ、ストリームを管理するために使用されるデータを含むデータ構造が形成される。図８を参照すると、各ストリームに対し、バッファ・ディレクトリ構造ＢＤＳが生成され、これが他のストリームのＢＤＳのリンク・リストに追加される。バッファ制御ブロック（ＢＣＢ）の連鎖が各ストリームに対して生成され、ストリームには各バッファに対して１つのＢＣＢが割当てられる。図示されるように、ＢＤＳ１がストリーム＃１に対して生成され、３つのＢＣＢすなわちＢＣＢ１－ＢＣＢ３に連鎖される。これらのＢＣＢはそれぞれ関連し、３つのバッファＢ１－Ｂ３に対するポインタがストリーム＃１に割当てられる。

【００４５】各ＢＤＳはストリームに割当てられるバッファの数、ストリームの状態、（分割ストリームにおける）バッファ内のレコードの数に関する情報、及び第１のフルのバッファのＢＣＢに対するポインタ２００、空バッファのＢＣＢに対するポインタ２０２、及び、リスト内の次のＢＤＳに対するポインタ２０４を記憶するための複数のフィールドを含む。ストリームの異なる状態には、実行中、停止、休止、及びプレロール（prerolled）或いはプライム（primed）がある。後者はオプションであり、ターゲット・デバイスがスタートする以前に、全てのバッファをソースから充填することによりクイック・スタートを提供する。各ＢＣＢはバッファ内のバイト数、分割ストリーム・バッファ内のレコードの数、バッファに対するポインタ２０６、連鎖内の次のＢＣＢに対するポインタ２０８、及び、ＢＣＢ連鎖の先頭及び末尾に対するポインタを含む情報を記憶する。

【００４６】ストリーム＃３及び＃４は図７の例を使用して、分割ストリームに対応するデータ構造のバッファリングを表す。ＢＤＳが分割ストリーム内の各データ・タイプに対応してセットアップされる。ＢＤＳ３はオーディオ・データ用に、またＢＤＳ４はビデオ・データ用にセットアップされる。バッファＢ１が３つのレコードにより充填されると、２つのレコード制御ブロックＲＣＢ１及びＲＣＢ２が生成され、充填されて、オーディオ・レコードＡＲ１及びＡＲ２のＢ１におけるオフセット及びオフセット長を示す。ＢＣＢ４はバッファＢ１に２つのレコードが存在することを示すようにセットされる。並行してＲＣＢ３が生成され、ビデオ・レコードＶＲ１のバッファＢ１におけるオフセット及びオフセット長を示す。ＢＣＢ７における記録フィールドの数は、そこに記憶される唯一のビデオ・レコードに対応して、１にセットされる。その他のＢＣＢ及びＲＣＢについても同様に生成され、分割ストリームに割当てられる残りのバッファに対応して充填される。また分割ストリームに対し、１つのＢＤＳがオーナに指定され、いずれか残りの１つがユーザと見なされる。リンク・リストには各ユーザＢＤＳから関連するオーナＢＤＳを指すポインタ２１０が含まれる。更に、次に示すポインタが存在する。それらはユーザＢＣＢからオーナＢＣＢを指すポインタ２１２、及び、ＢＣＢからそれに連鎖されるＲＣＢを指

すポインタ２１４である。

【００４７】ＳＳＭ１１４はストリーム・マネージャ・ヘルパ（ＳＭＨ）を使用することにより、ストリーム・ハンドラから呼出される。図９を参照すると、ステップ２２０で、マネージャ１１４がバッファ或いはレコードを獲得或いは返却するためにSMH.NOTIFY呼出を受信すると、ステップ２２２で呼出パラメータが確認され、ステップ２２４で呼出者がソース・ストリーム・ハンドラか、或いはターゲット・ストリーム・ハンドラであるかを判断する。ソース・ストリーム・ハンドラである場合、ステップ２２６で、ユーザ或いはアプリケーションがバッファを供給しているかどうかを判断する。そうである場合、ステップ２２８で GiveBufルーチンを呼出し、このルーチンの復帰に際し、ステップ２３０は呼出者に復帰する。ステップ２２６が否定の場合、ステップ２３２はフルのバッファが返却されているかを判断する。そうである場合、ステップ２３４でReturnFullルーチンを呼出し、そのルーチンの復帰に際しフルのバッファが返却される場合には、ステップ２３０を通じて復帰するか、或いは空バッファに対する要求が生じているか判断するためにステップ２３６が実行される。ステップ２３６は、ステップ２３２でフルのバッファに対する要求が出されていないと判断されると、常に実行される。ステップ２３６が肯定の場合、ステップ２３８でGetEmptyルーチンが呼出され、ステップ２３０を介して復帰する。

【００４８】ステップ２２４で呼出者がソース・ストリーム・ハンドラではないと示される場合には、呼出者はターゲットであり、ステップ２４０で空バッファが返却されているかを判断する。返却されている場合、ステップ２４２で ReturnEmptyルーチンを呼出し、ステップ２４４で要求がフルのバッファに対するものであるかを判断する。そうでない場合、ステップ２３０を介して復帰する。そうである場合は、ステップ２４６で GetFullルーチンを呼出し、その後、ステップ２３０を介して復帰する。ステップ２４０が否定の場合、ステップ２４８でストリームの終わりに達したかがチェックされる。達してない場合、ステップ２４４に移行し前述のように処理される。一方、達した場合には、ステップ２５０でバッファをリセットし、ステップ２５２でアプリケーションにストリームの終わりを報告し、その後、ステップ２４４に移行する。

【００４９】図１０は GiveBufルーチン２２８を示し、ステップ２５４は提供すべき残りのバッファが存在するかをチェックする。存在しない場合、ステップ２５６で復帰する。そうしたバッファが存在する場合は、ステップ２５７で空ポインタがエンプティＢＣＢを指示するかをチェックする。指示される場合、ステップ２５８でMapUserBufferルーチンを呼出す。その後、ステップ２６０でエラーをチェックする。エラーが発生していない場合、ステップ２６２でバッファの属性を保管し、ステップ２６４でストリームの終わりに達したかが確認される。達していない場合、ステップ２６６でターゲット待機中フラグがセットされているかがチェックされる。セットされていない場合、ステップ２５４にループが形成される。ステップ２５７が否定の場合、ステップ２６８でソース待機中フラグをセットし、ステップ２７０でCheckPrerollルーチンを呼出す。ステップ２７２ではエラーをチェックする。エラーが存在しない場合、制御はステップ２５４に移行する。ステップ２７２でエラーが通知される場合、ステップ２７４でバッファ過剰を示すエラー・フラグをセットし、ステップ２５６により復帰する。ステップ２６４が肯定の場合、ステップ２７６で Check-Prerollルーチンを呼出し、ステップ２５４に移行する。ステップ２６６が肯定の場合には、ステップ２７８で、ターゲット・ストリーム・ハンドラを起動するためにヘルパ・コマンドを送信する。

【００５０】図１１を参照すると、フルのバッファを獲得するためにReturnFullルーチンが呼出されると、最初にステップ２８０で、使用可能なフルのバッファが存在するかがチェックされる。存在しない場合、ステップ２８２で復帰する。フルのバッファが存在する場合、ステップ２８４でフルのバッファのアドレスに対応するＢＣＢの連鎖を探索し、ステップ２８６でフルのバッファが見い出されたか、及びソースがそれを所有するかがチェックされる。否定の場合、ステップ２８８で無効なバッファの返却を示すエラー・フラグをセットし、ステップ２８２で復帰する。ステップ２８６が肯定の場合、ステップ２９０でバッファ或いはレコードが返却されているかが判断される。バッファの場合には、ステップ２９２でバッファの属性を保管し、ステップ２９４で保留の放棄ストップが存在するかがチェックされる。存在する場合、ステップ３００でCheckStopPendingルーチンを呼出す。ステップ２９６はステップ３００或いはステップ２９４に続き、ストリームの終わりに達したかを判断する。達した場合、ステップ３０２でCheckPrerollルーチンを呼び出す。ストリームの終わりに達していない場合は、ステップ２９８で、ターゲット・ストリーム・ハンドラが起動されるのを待機しているかがチェックされる。そうである場合は、ステップ２９９でターゲットを起動するためにSHC.START ルーチンを呼出す。ステップ２９８が否定の場合、ステップ２８０へループが形成される。

【００５１】ステップ２９０でレコードが返却されると判断されると、ステップ３０４はレコードがバッファに存在するかを確認する。ステップ３０６はレコード・ポインタがｏｋであるかを判断する。そうでない場合、ステップ３０８で無効なレコードが返却されたことを示すエラー・フラグをセットし、その後、ステップ２８２で復帰する。ステップ２０６が肯定の場合、ステップ３１

０でレコードの属性を保管し、ＲＣＢを割当て、それをＲＣＢリストに追加する。次にステップ３１２でストリームの終わり及びバッファ状態に対応する最終レコードをチェックし、ステップ２９４に移行する。

【００５２】図１２に示されるGetEmptyルーチンの開始に際し、ステップ３１４は獲得すべきバッファが更に存在するかを判断する。存在しない場合、ステップ３１６で復帰する。存在する場合は、ステップ３１８で空ポインタがエンプティＢＣＢを指示するかをチェックする。そうである場合、ステップ３２０でバッファの属性を獲得し、ステップ３１４にループして戻る。ステップ３１８が否定の場合、ステップ３２２で使用可能なバッファが存在しないことを示すソース待機中フラグをターン・オンする。次にステップ３２４で、CheckPrerollルーチンを呼出す。ステップ３２６でエラーをチェックする。エラーが存在しない場合、ステップ３１４へのループ３１４が形成される。エラーが存在する場合、使用可能なバッファが存在しないことを示すためにエラー・フラグがセットされ、ステップ３１６で復帰する。

【００５３】図１３を参照すると、ReturnEmpty ルーチンはステップ３３０で開始され、ここで返却すべきバッファが更に存在するかがチェックされる。存在しない場合、ステップ３３２で復帰する。バッファが存在する場合、ステップ３３４でバッファ・アドレスに対応するバッファＢＣＢの連鎖を探索する。ステップ３３６では、バッファが見い出されて、ターゲット・ストリーム・ハンドラがそれを所有するかが確認される。そうでない場合には、ステップ３３８で無効なバッファが返却されたことを示すエラー・フラグがセットされる。ステップ３３６が肯定の場合、ステップ３４０で、レコード或いはバッファのどちらが返却されるかを判断する。バッファの場合、ステップ３４２でバッファを空と記し、ステップ３４６でソース・ストリーム・ハンドラが待機中であるかを確認する。そうでない場合は、ステップ３４８をスキップする。待機中の場合は、ステップ３４８でソースに対しSHC.START ルーチンを呼出す。ステップ３４４が肯定の場合、次にステップ３５０はCheckStopPendingルーチンを呼出す。ステップ３４８或いは３５０の後にはステップ３３０が続く。ステップ３４０でレコードが返却されると判断されると、ステップ３５２はＲＣＢリストから関連するＲＣＢを除去し、ステップ３５４で全てのレコードが空状態に戻されたかを判断する。そうである場合、ステップ３５６でバッファを空と記し、次にステップ３４４がステップ３５４或いは３５６に続く。

【００５４】GetFull ルーチン（図１４）では、ステップ３５８で更に獲得するバッファが存在するかを判断する。存在しない場合、３６０で復帰する。存在する場合、ステップ３６２でフルのポインタがフルのバッファを指示するかを確認する。否定の場合、使用可能なバッファは存在せず、ステップ３７０でターゲット待機中フラグをセットし、３６０を介して復帰する。３６２が肯定の場合、ステップ３６４でストリームのタイプがノーマル（統一型）か、或いはインタリーブドかを判断する。統一型の場合、ステップ３６６でバッファの属性を獲得し、ステップ３６８で最終バッファの時にストリームの終わり状態（ＥＯＳ）をセットし、次に３６０を介して復帰する。インタリーブド或いは分割ストリームに対しては、ステップ３７２でバッファ内のフルのレコードを探索する。３７４のチェックにより発見される場合、ステップ３７６でレコードの属性を獲得し、ステップ３７８でバッファ内の最終レコードをチェックし、その後３６０を介して復帰する。

【００５５】図１５を参照すると、CheckStopPendingルーチンはステップ３８０で全てのバッファが返却されたかを確認する。そうでない場合、ステップ３８４で復帰する。返却された場合は、ステップ３８６でSMH.ReportEventルーチンを呼出し、"ストリームの停止"を報告し、３８４で復帰する。

【００５６】図１６はCHECK.PREROLL ルーチンのオペレーションを示し、ステップ３８８はストリームがプレロール或いはプライムされたか、且つｓｙｎｃグループに含まれないかを判断する。否定の場合、ステップ３９０でストリームがマスタであり、全てのスレーブがプレロールされたかをチェックする。否定の場合、３９２を介して復帰する。ステップ３８８及び３９０の結果が肯定の場合は、ステップ３９４でSMH.ReportEvent ルーチンを呼出し、"ストリーム・プレロール"を報告し、３９２を介して復帰する。

【００５７】図１７はリング０及びリング３において使用可能な呼出であるSMH.ReportEvent ルーチンを示す。ステップ３９６はイベント・キューがオーバーフローしたかをチェックする。オーバーフローした場合、ステップ３９７でエラー事象“イベント・キュー・オーバーフロー”を報告し、ステップ３９８に移行する。また、イベント・キューのオーバーフローが生じていない場合は、ステップ３９８でその事象を特定の処理に対するイベント・キューに加え、ステップ４００でリング３のイベント・スレッドをブロック解除し、ステップ４０２で復帰する。

【００５８】図１８はMapUserBuffer ルーチンのオペレーションを示し、ステップ４０４ではバッファが既にロックされているか、すなわち、バッファが既に自己のリスト内に存在するかを確認するために探索する。バッファがロックされていれば、ステップ４０６はステップ４１６に移行し、そのバッファに対応するＢＣＢを新たなＢＣＢリストにコピーし、次に４１８で復帰する。バッファがロックされていない場合は、ステップ４０８でバッファをロックしようと試行する。ステップ４１０では、後者のステップが成功したかを判断する。成功した場合、ステップ４２０でフィールド情報をそのＢＣＢに

追加し、更にＢＣＢをバッファ・キューの末尾に追加し、その後４１８で復帰する。４１０が不成功の場合、ステップ４１２で新たにロックされたバッファを割当て、ステップ４１４でバッファ・データを新たなバッファにコピーし、その後ステップ４２０及び４１８を実行する。

【００５９】図１９では、事象チェックがルーチンの目的のポイントで実施され、ステップ４２２ではスレッドをブロックする。スレッドがブロック解除されると、EventRouter ルーチンがステップ４２４で呼出され、このルーチンは最初にステップ４２６で事象に対応するイベント・キューをチェックし、次にステップ４２８で事象が取り出される（ dequeue）。ステップ４３０では事象を取り扱うアプリケーション・イベント・ルーチンが呼出される。次にステップ４３２で、更に処理する事象が存在するかがチェックされ、存在する場合は、ステップ４２８に戻る。全ての事象が処理されると、ステップ４３４でスレッドがブロックされる。

【００６０】図２０を参照すると、アプリケーション・プログラムがストリームを生成すると、SpiCreateStream API呼出が実施され、制御はsync/streamマネージャ１１４に移り、マネージャ１１４はステップ４３６でSHC.CREATEコマンドを使用して、ソース・ストリーム・ハンドラを呼出す。次にステップ４３８で、SHC.CREATEを使用して、ターゲット・ストリーム・ハンドラが生成される。ステップ４４０ではストリーミング・プロトコル制御ブロック（図示せず）にリストされるストリーミング・パラメータが獲得され、パラメータにはバッファ数及びサイズ、データ・ストリーム・タイプ、バッファ当たりの最大レコード数、一定のデータ・ストリームを維持するために必要な最小バッファ数、ソース・ストリーム・ハンドラを起動するために必要な空バッファ数、ターゲット・ストリーム・ハンドラを起動するために必要なフル・バッファ数などが含まれる。次にステップ４４２で、ソース・ストリーム・ハンドラにストリーム・パラメータを通知し、ステップ４４４で、ターゲット・ストリーム・ハンドラにストリーム・パラメータを通知する。ステップ４４６ではストリームをセットアップし、ストリーム・バッファを割当てる。その後、ステップ４４８でアプリケーションに復帰する。

【００６１】図２１を参照すると、アプリケーション・プログラムがストリーミング開始呼出を発行すると、マネージャ１１４はSpiStartStream API呼出を受信し、ステップ４５０がSpiStartStreamルーチンを実行し、これは最初にステップ４５２でストリームが休止状態かをチェックする。休止状態でない場合、ステップ４５４でストリームがプレロールされているかを判断する。されていない場合、ステップ４５６で、SHC.START コマンドによりソース・ストリーム・ハンドラを呼出し、ソース・スレッドをブロック解除する。次にステップ４５８で、ストリームがｓｙｎｃグループにおけるスレーブ・ストリームかをチェックする。そうでない場合、ステップ４６０でアプリケーション・プログラムに復帰する。そうである場合は、ステップ４６２でステップ４５２に分岐し、各スレーブ・ストリームに対応する処理を繰り返す。ステップ４５２が肯定の場合、ステップ４６４でSHC.START コマンドにより、ソース・ストリーム・ハンドラを呼出してスレッドを解除し、次にステップ４６６で、ターゲット・ストリーム・ハンドラを呼出し、ブロック解除する。ステップ４６６はまた、ステップ４５４の肯定結果にも続く。次に、ステップ４５８がステップ４６６に続いて実施される。

【００６２】図２２はイベント・データ構造を表す。各処理に対し、イベント・キューを指示する処理制御ブロックＰＣＢが生成される。図示のように２つのＰＣＢすなわちＰＣＢ１及びＰＣＢ２が存在し、それぞれ処理＃１及び＃２に対応してセットアップされる。各ＰＣＢはプロセスＩＤ、イベント・スレッドＩＤ、関連するイベント・キューの先頭を指示するポインタ４７２を含む。イベント・キューはヘッダ４７４を含み、各ヘッダはフラグ及びユーザ・イベント制御ブロックＥＶＣＢ４７８を指示するポインタを含む。こうしたブロックは拡大して示されるように、イベント・タイプ、イベント・フラグ、ストリーム・ハンドラＩＤ、そのステータス、及びユーザ・パラメータを含む。イベント・データ構造内の情報はイベント・ルータ・ルーチン（図１９）により使用される。

【００６３】図２３は SpiEnableルーチン４８０のオペレーションを表し、ステップ４８２では呼出すストリーム・ハンドラを決定し、ステップ４８４でステップ４８２で識別されたソース或いはターゲット・ハンドラを呼出す。ステップ４８４の後にはスキップ４８６が続く。

【００６４】図２４はＤＬＬの初期化時に生成されるアプリケーションからの最初のＳｐｉＡＰＩ呼出により実行される内容を表す。ステップ４８８でEventRouterルーチンを呼出すことによりイベント・スレッドを生成する。ステップ４９０でイベント・スレッドの優先度をセットし（リング３または０）、ステップ４９２でsync/streamマネージャを呼出し、ステップ４９４で復帰する。

【００６５】図２５は時間キュー （cue）・ポイント事象と称される事象の例を示し、ストリーム・ハンドラ特にターゲット・ストリーム・ハンドラ内に配置される。ステップ４９６は現行の時間を決定する。次にステップ４９８は、こうした時間において発生する事象に対応するイベント・キュー或いはリストを探索する。ステップ５００は時間の一致をチェックする。一致が存在しない場合、ステップ５０４で復帰する。各一致に対し、ステップ５０２で SMH.REPORTEVENTの呼出によりその事象を報告する。

【００６６】以下本願発明の内容を要約して述べる。
【００６７】本発明はいくつかの独自の特徴を有し、これらは新規なものであり、従来、アプリケーション或いはオペレーティング・システム・ソフトウェアにおけるマルチメディア・データ伝送には組み込まれていなかった。本発明の独自の特徴としては次の点が含まれる。すなわち、アプリケーション・バッファリング・オプションによる中央バッファ管理；２レベルのデータ・ストリーム・ハンドラのサポート；インタリーブド・データ・ストリームのサポート；及び、データ・ストリーム事象の検出及び通知である。これらの独自の特徴は、存在する物理デバイス・ドライバを超越して完全にデバイスに独立な形式により実施される。この解決においては、更に、データ・タイプはマルチメディア・システム拡張におけるモジュールに依存しない。
【００６８】本発明の一般化したデータ・ストリーミング機構はデバイス独立型のマルチメディア・データ伝送を提供し、ＩＢＭのバージョン２．０のオペレーティング・システム／２（ＯＳ／２）に対するマルチメディア拡張として設計される。この機構は実時間という条件を具備しつつ、記憶及び／或いは検索のためのデータの連続的な転送を支援する。例えば、大量のデジタル化オーディオ・データ・ファイルはハード・ファイルから読出され、データを連続的にアナログ増幅回路を有するデジタルーアナログ変換器（ＤＡＣ）装置に転送することにより演奏されねばならない。
【００６９】データ・ストリーミングの実施例の次の４つの主な特徴は、本発明の中心を成すものである。また、これらはマルチメディア・コンピュータ・システムにおける重要な貢献を代表するものであり、前述のセクションで述べられた全ての問題を解決するものである。
【００７０】アプリケーション・バッファリング・オプションによる中央バッファ管理
【００７１】データ・ストリーム・リソース管理は中央バッファ・プールに対する／からのバッファ割当を制御し、複数のストリーム・ハンドラの間でバッファを共用し、また動的なバッファのロック／ロック解除を制御する。更にデータ・ストリームは、アプリケーションにより割当てられ管理されるバッファを使用し、特にこれらはアプリケーションが、ストリームを介して転送されるデータの実時間生成を制御している場合に使用される。
【００７２】一連のバッファ管理ルーチンは、データ・ストリームが生成される時に、割当てられたバッファ・プールからバッファを獲得しまた返却する機能を提供する。これらのバッファ用のメモリは割当てられ、この時にロック・ダウンされる。バッファのロック・ダウンは、データのストリーミング中に、システムがバッファを含むメモリをスワップ・アウトしないことを保証するために必要である。
【００７３】処理されるデータが連続的なストリームとして処理されることを保証するために、ストリームに関連するストリーム・プロトコル制御ブロックがセットアップされ、これはバッファを割当てるために使用される情報（すなわちバッファ数、バッファ・サイズ、及びストリーム・ハンドラを起動する時を示す値）を含む。ストリームされるデータのタイプによって、バッファの量及びサイズを最適に設定しなければならないためこの情報は重要である。なぜなら、データの連続的なストリーミングの継続、及びシステム・リソースが過度に（すなわち実際に必要なメモリ量以上にメモリを割当てる）使用されないことを保証する必要があるためである。メモリの割当と共に、ストリーム・ハンドラを起動するために使用される値は、ソース・ストリーム・ハンドラにより十分なバッファが充填され、その結果ターゲット・ストリーム・ハンドラが起動可能であり、かつ、実時間データ・ストリーミングを発生するために十分なデータがバッファされていること、を保証する。
【００７４】中央バッファ管理ルーチンは、ソース・ストリーム・ハンドラがバッファ・プールから空バッファを獲得し、入力装置からのデータ（例えばハード・ディスク上のファイルからのデータ）を充填し、それによってフルになったバッファをバッファ・プールに返却することを許可する。ターゲット・ストリーム・ハンドラはバッファ・プールからフルになったバッファを獲得し、このバッファを出力装置（例えばスピーカにより演奏されるオーディオ装置へデータを転送する）に送信し、その結果空になったバッファをバッファ・プールに返却する。
【００７５】バッファ管理ルーチンの独自の特徴は、バッファ内でレコードを扱う機能である。この機能によりインタリーブド・データは最小のデータ転送により処理される。データは一度だけバッファに転送され、続くアクセスは個々のデータ（すなわちオーディオ及びビデオ・データなど）を含むバッファ内のレコードを介して行われる。
【００７６】バッファ管理ルーチンの他の独自の特徴は、ソース・ストリーム・ハンドラがその固有のバッファをバッファ・プールに提供することを可能とし、その後、このバッファは要求により、ターゲット・ストリーム・ハンドラに提供される。バッファが提供されると、これらはその時点でメモリにロック・ダウンされ、これによりスワップ・アウトされることはなくなり、また、データ・ストリームの期間中に、全システム・メモリ・リソースを消費することはない。
【００７７】次に典型的なストリーム・ハンドラがファイルからデータを読出し、ストリームをオーディオ出力装置に転送するためのシナリオを示す。
【００７８】１．アプリケーションがデータ・ストリームを生成する。バッファが割当てられ、この時ロック・ダウンされる。

２．アプリケーションがデータ・ストリーミングを開始する。
３．ソース・ストリーム・ハンドラが
－空バッファを獲得する。
－入力装置からバッファを充填する。
－フルのバッファを返却する。
－データの終わりに遭遇するまで、上記３ステップをループする。
４．ターゲット・ストリーム・ハンドラが
－フルのバッファを獲得する。
－バッファを出力装置に排出する。
－空バッファを返却する。
－データの終わりに遭遇するまで、上記３ステップをループする。

【００７９】こうした一連のバッファ管理ルーチンは、ソース・ストリーム・ハンドラからターゲット・ストリーム・ハンドラへの連続的なデータ・フローの発生を保証する機能を提供する。

【００８０】２レベルのデータ・ストリーム・ハンドラの支持

【００８１】異なる保護レベルのデータ・ストリーム・ハンドラに対し、同一のシステム・サービスを提供し、システム・ハンドラがＯＳ／２物理デバイス・ドライバ（保護リング０）或いはＯＳ／２動的リンク・ライブラリ（保護リング３）として開発されることを可能とする。

【００８２】ＯＳ／２アーキテクチャは８０２８６を基本とするインテル・ファミリのプロセッサを支持し、これには８０３８６ＤＸ、８０３８６ＳＸ、及び８０４８６などが含まれる。これらのプロセッサはリング０から３までの一連の特権レベル或いはリングの提供を基本とする保護機構を共有する。各レベルにおいて、異なるオペレーションが、システムの保全性に影響を与える相対的リスクに準じて許可される。特権レベル０或いは"リング０"では、全てのオペレーションは許可される。このレベルは典型的にはシステム・コードの実行用に確保される。アプリケーションはリング３で実行され、特権命令の実行の試行はトラップを生成し、これはリング０のオペレーティング・システムにより扱われ、典型的にはアプリケーションの終了に帰結する。

【００８３】リング０のソフトウェアの開発は、リング３のソフトウェアの開発に比較して困難であり、またリスクを伴う。これはリング０で活動状態のシステム保護機構の相対的な欠如による。リング０のコードのモジュール開発、テスト、及びそこからエラーを除去するためには、カーネル・デバッガを含む特殊なツールが必要となる。その結果、リング０のコードの特殊な性質を必要としない機能、例えばハードウェア割込みのコンテキストにおいて実行される機能などは、典型的にはリング３で実行されるように作成される。リング３におけるデバッグはより簡単であり、リング３のコードにより生成されるルーチン・エラーは、リング３で実行可能な全てのコードに対して提供される保護機能により、一般には他の処理に影響を及ぼすことはない。

【００８４】通常、開発者は、リング０の特権の持つ特殊な性質により目的の機能が提供される時以外は、全てのコードをリング３で実行可能なように開発することを好む。しかしながら、ＯＳ／２のリング０のモジュールの開発においては、使用可能なシステム・サービスの数及び柔軟性は非常に制限されている。リング０の実行に対応してロード可能なモジュールのタイプは、唯一、物理デバイス・ドライバ（ＰＤＤ）である。ＰＤＤにとって唯一使用可能なシステム・サービスは、ＯＳ／２カーネルにより提供されるデバイス・ヘルプ（ＤｅｖＨｌｐ）機能である。これらの同一のＤｅｖＨｌｐインタフェースはリング３のモジュールでは使用できず、全体的に異なるセットのシステム・サービス・インタフェースを使用してプログラムされねばならない。

【００８５】ここでオペレーティング・システムの拡張が導入される時に、常に問題が生ずる。新たなサービスがリング３に限り提供される場合、広範囲に及ぶデバイス制御機能が妨げられる。なぜなら新たなサービスを呼出すモジュールについても、特権レベル０命令を実行することはできなかったり、或いはハードウェア割込みコンテキストにおいて実行できないからである。新たなサービスがリング０に限り提供される場合、全ての開発者は、彼らが特権レベル０の命令をアクセスする必要がある無しに関わらず、リング０のモジュールを作成する困難を被ることになる。

【００８６】好適な実施例としては、データ・ストリーム制御サービスを提供するＯＳ／２の拡張である。これらのサービスはＯＳ／２のＤＬＬ及びＰＤＤの対により構成されるSync/Streamマネージャにより提供される。データ・ストリームはSync/Stream マネージャにより提供されるサービスを呼び出す１つ以上のストリーム・ハンドラにより制御される。

【００８７】ストリーム・ハンドラの開発者に対して最大の柔軟性を提供し、リング０のコードの急速な拡張を阻止するために、Sync/Stream マネージャにより同一のサービスがリング０及びリング３の両方において提供される。これはすなわち、リング０の特権を厳格に要求するこれらのストリーム・ハンドラだけが、ＯＳ／２のＰＤＤとして作成され、他の全てはリング３で実行されるＯＳ／２のＤＬＬとして作成されることを意味する。両方の特権レベルにおいて同一のインタフェースが使用可能なため、ストリーム・ハンドラの開発者はデータ・ストリームを制御するために、１つのセットのインタフェースだけを使用して、コードを作成できる。

【００８８】Sync/Stream マネージャは重複するサービス・ルーチンを生成することなしに、リング０及びリン

グ３の両方のモジュールに対し、同一のサービスを提供する。各機能に対して２つの入力ポイントが提供され、一方はリング３（Sync/Stream マネージャＤＬＬ対応）であり、他方はリング０（Sync/Streamマネージャ ＰＤＤ対応）である。これらの入力ポイントの各々は共通のサブルーチンにベクトル指向される。共通のコードが、呼出される特定の機能に最適となるように、リング０或いはリング３で実行される。

【００８９】インタリーブド・データ・ストリームの支持

【００９０】この機構により、単一のデータ・ストリーム・ソースは、データを複数のターゲットに転送することが可能となる。これは分割ストリームと称される。例えば、ある装置はオーディオ、イメージ、及びテキストのデータ・タイプを、単一のフォーマットされたデータ・トラックにインタリーブする。分割ストリーム・ハンドラは、単一のインタリーブド・データ・バッファから各データ・タイプを抽出し、各データ・タイプを独立に転送し、各ストリームを適切なターゲット・デバイスに引き渡すために使用される。例えば、CD-ROM XA はウェーブオーディオ、ビデオ・データ及びテキスト・データを同一のデータ・バッファ内にインタリーブできる。分割ストリーム・ハンドラは、データをCD-ROM XA 装置用の複数のストリームに分割するために使用される。分割ストリーム・ハンドラにとって、この結合されたデータを１つのバッファに読出し、次にデータをコピーすることなく、ビデオ・データをビデオ・ストリームに挿入し、またウェーブオーディオ・データをウェーブオーディオ・ストリームに挿入することが、より効率的である。ウェーブオーディオ・データ・ストリームは次にウェーブオーディオ・ストリーム・ハンドラにより、またビデオ・ストリームはビデオ・ストリーム・ハンドラにより消費される。ウェーブオーディオ・ストリーム・ハンドラはデータ・ストリームのオーディオ部分だけを認識し、またビデオ・ストリーム・ハンドラはストリームのビデオ部分だけを認識する。

【００９１】データ・ストリームは複数の分割ストリームに分割される。分割の数はシステムのリソースによってのみ制限される。別々に分割されたストリームの各々は分割ストリームである。異なるデータを分析し決定する作業は、ソース・ストリーム・ハンドラの役割であり、実際には空バッファをソース・デバイスからのデータにより充填することを行う。ソース・ストリーム・ハンドラは、どのデータがどのストリームに所属するかを決定し、次に、ポインタをバッファの個々の"レコード"に返却しなければならない。各レコードは、ある"分割"ストリームに特定なデータにより充填されるバッファの一部に相当する。これらのレコードはSync/Stream マネージャに返却され、どのストリームに所属すべきかを示す。Sync/Stream マネージャは、次に、バッファを共用する分割ストリームの１つに対応して設けられたそれぞれのバッファ／レコード・キューに、これらのレコードをキュー待機させる。

【００９２】このタイプの分割ストリームは以下に示すようにセットアップされる。複数のストリームが生成されなければならず、各々はSync/Stream マネージャに対するSpiCreateStream API 呼出により実行される。これらのＡＰＩ呼出の第１は、通常のストリームの生成呼出であり、hstreamBufパラメータはNULLにセットされる。第１のストリームのバッファを共用する必要のある他の続くSpiCreateStreamAPI呼出のそれぞれは、hstreamBuf パラメータにおけるそのストリームの制御をパスしなければならない。これはSync/Stream マネージャにより使用される機構であり、これらの分割ストリームによって第１のストリームのバッファを共用するためのものである。ソース・ストリーム・ハンドラはまた、SHC_CREATE呼出によりこれらのパラメータを受信し、この種のタイプのストリーミングを支持することが必要である。SPCBにおける SPCBBUF_INTERLEAVEDフラグはソース・ストリーム・ハンドラによりセットされ、分割ストリーミングを実行可能かどうかを示す。実行できない場合、SpiCreateStream がそのバッファを使用する試みは拒否される。分割ストリーム機構は１つのソース及び複数のターゲットに対応して機能する。

【００９３】データ・ストリーム事象の検出及び通知

【００９４】ストリーム事象はデータ・ストリームにおける状態の特定の変更を識別する。データ・ストリーム・ハンドラは特定の事象を検出し、その事象を無視するか、或いはその事象が発生したことを通知する。この通知は別のストリーム・ハンドラ或いはアプリケーション・プログラムにより受信される。

【００９５】アプリケーションはデータ・ストリーミング・ルーチンに対し、（イネーブル／ディスイネーブル事象を介し）どの事象が検出されるべきかを知らせることができる。これらの事象は典型的にはエラー検出の通知、ストリーム・キュー時間事象、及びデータ・キュー事象などである。また、ストリーム・ハンドラはデータ・ストリームの実時間性を維持するために実行される特定の活動を保証するデータ・ストリームの状態を通知される必要がある。このために、これらの事象はアプリケーション或いはストリーム・ハンドラ・イベント・ルーチンに適時（実時間で）送信されて、最小時間が経過する以前に適切な活動が実行される必要がある。この事象の報告は、ストリーム・ハンドラがリング０或いはリング３で実行されているという事実によって若干複雑になる。

【００９６】事象に関するリング３のストリーム・ハンドラからリング３のストリーム・ハンドラ或いはアプリケーションへの報告は、登録されたイベント・ルーチンがスレッドによる事象報告のコンテキストにおいて呼出

されるという点で、直接的である。しかしながら、リング０のストリーム・ハンドラがリング３のストリーム・ハンドラ或いはアプリケーションに対して事象を報告する必要が生じた場合には、リング０の割込み時間中にストリーム・ハンドラが事象を検出した場合、この時間内に直接リング３のイベント・ルーチンを呼出す方法が存在しないという点で問題が生ずる。この問題を解決するために、最小のタスク切替えオーバーヘッドを有し、実時間環境において事象を十分な速度で報告できるようにディスパッチされる新たなタイム・クリティカル・スレッドが生成された。この新たなスレッドは実行中は割込み不可である。各アプリケーション処理に対し、この新たなスレッドが生成され、報告される事象をモニタする。事象報告するための事象シーケンスは以下のようである。

【００９７】１．事象が割込み時間において、リング０のストリーム・ハンドラにより検出される。

２．事象はイベント・キューに配置され、新たなタイム・クリティカル・スレッドがブロック解除されて実行される。これはリング３のアプリケーションのコンテキストにおいて実行される。

３．タイム・クリティカル・スレッドはアプリケーションのコンテキストにおいて制御を獲得し、イベント・キューから事象を取り出し、アプリケーション・イベント・ルーチンを呼出す。

【００９８】この事象シーケンスは最小時間により実行され、それによりアプリケーションは活動を実時間で実行でき、また見た目にもより優れたアプリケーションが指向される。

【００９９】当業者においては、本発明の精神及び範中を逸脱することなく、構成及び詳細における多くの変更が可能であることが理解されよう。

【０１００】

【発明の効果】以上説明したように、本発明によれば、マルチタスキング・オペレーティング・システムの制御の下で、実時間において連続的に大量のデータを伝送することを可能とするマルチメディア・システムが提供できる。

【図面の簡単な説明】

【図１】本発明を実施するデータ処理システムのブロック図である。

【図２】図１のシステムにおいて実施されるsync/streamサブシステム・アーキテクチャのブロック図である。

【図３】データ・ストリーミングの一般化モデルを表すブロック図である。

【図４】ソース・ストリーム・ハンドラ・オペレーションの流れ図である。

【図５】ターゲット・ストリーム・ハンドラ・オペレーションの流れ図である。

【図６】ターゲット・ストリーム・ハンドリング・オペレーションの流れ図である。

【図７】統一及び分割データ・バッファリングを表す図である。

【図８】バッファリング・データ構造を示す図である。

【図９】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図１０】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図１１】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図１２】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図１３】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図１４】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図１５】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図１６】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図１７】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図１８】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図１９】sync/stream バッファ管理オペレーションをインターロックする流れ図である。

【図２０】sync/stream マネージャ・ストリーム生成オペレーションの流れ図である。

【図２１】sync/stream マネージャ・ストリーム開始オペレーションの流れ図である。

【図２２】事象データ構造を示す図である。

【図２３】事象許可オペレーションの流れ図である。

【図２４】オペレーションがアプリケーション・プログラムからsync/streamマネージャに移行する様子を示す流れ図である。

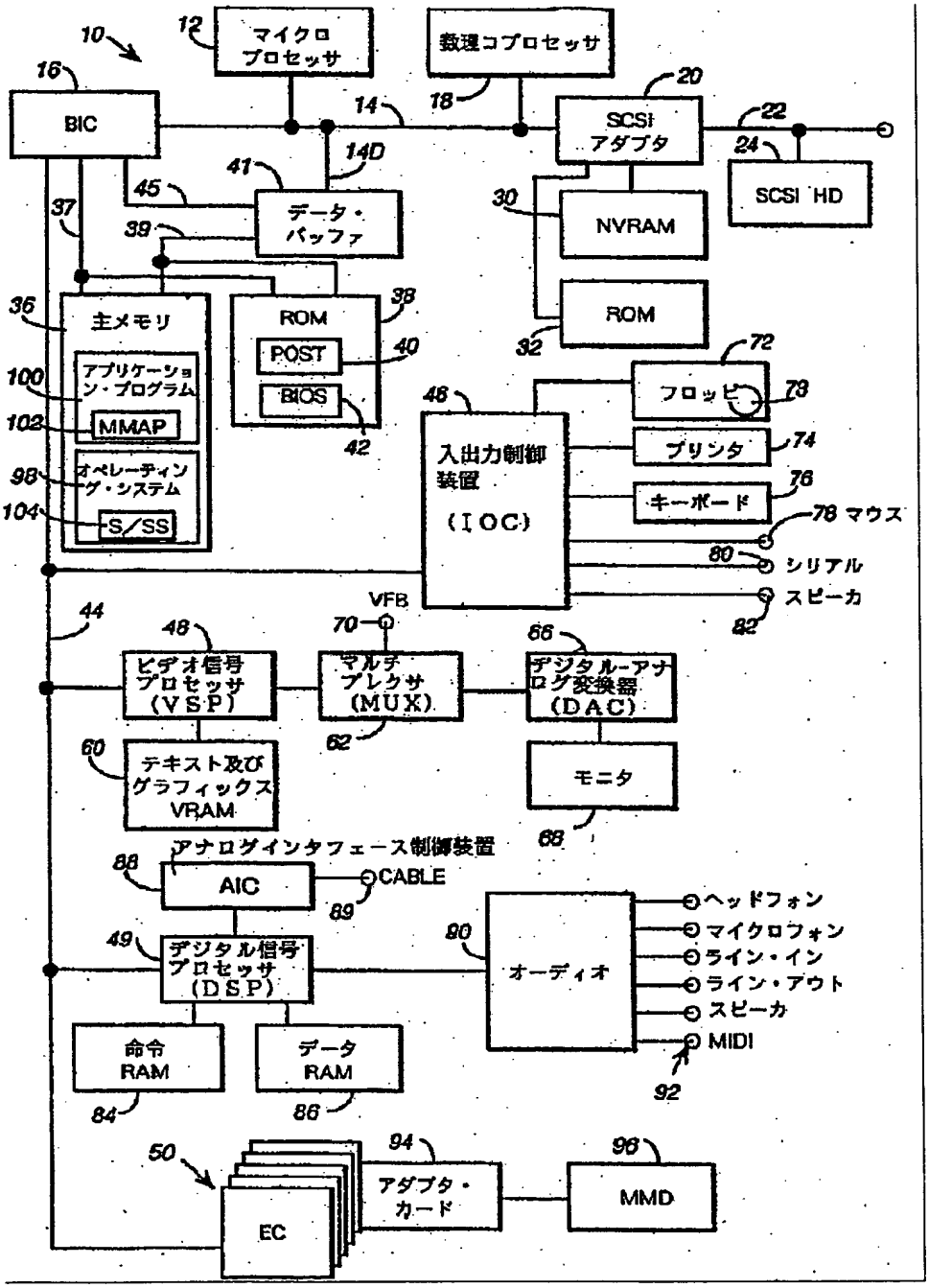
【図２５】事象検出オペレーションの例を示す流れ図である。

【符号の説明】

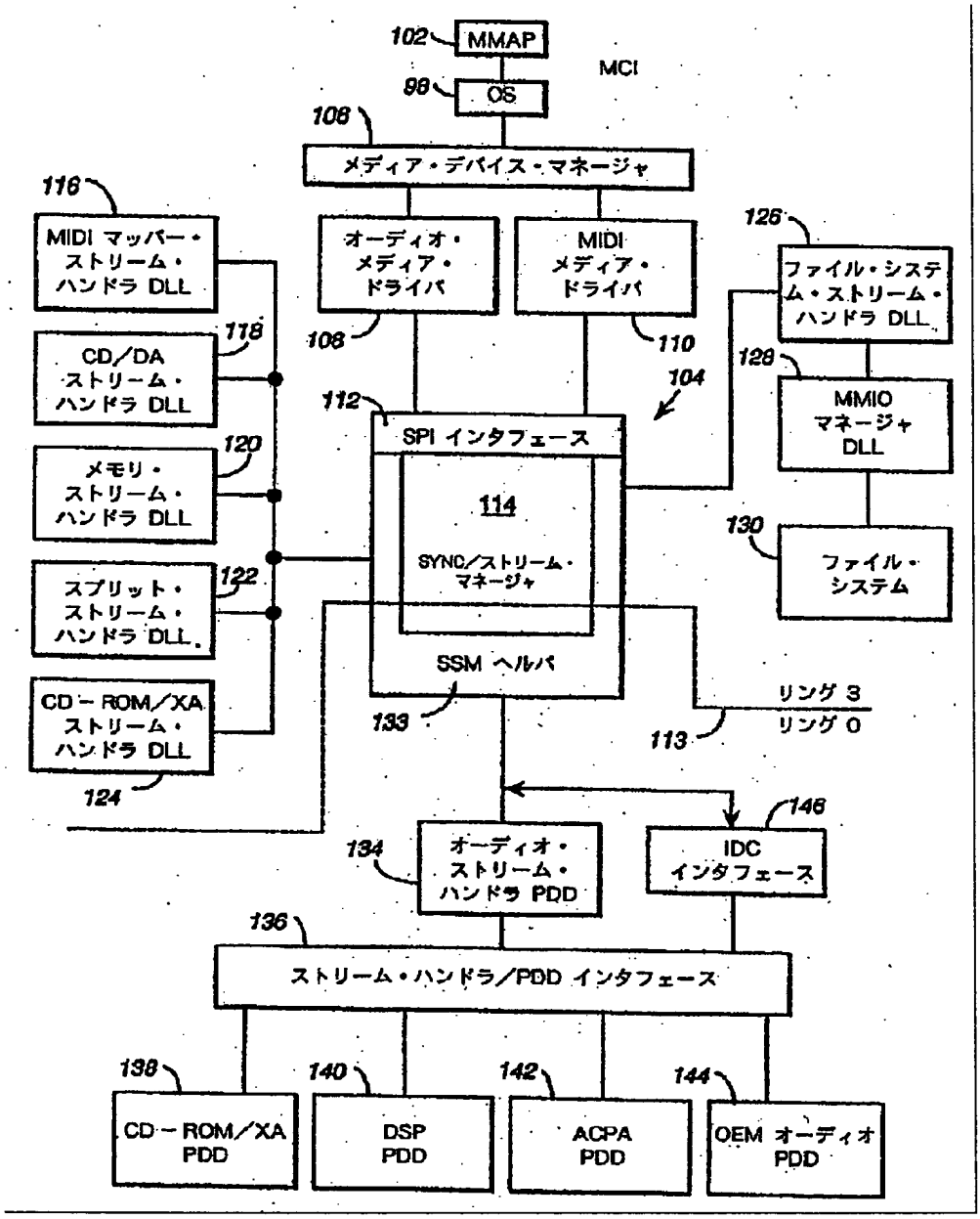
２２・・・ＳＣＳＩバス
２４・・・ＳＣＳＩハード・ドライブ（HD）
３７・・・制御バス
３９・・・データ・バス
４１・・・データ・バッファ
４８・・・ビデオ信号プロセッサ（ＶＳＰ）
４９・・・デジタル信号プロセッサ（ＤＳＰ）
６０・・・ビデオＲＡＭ（VRAM）
６２・・・マルチプレクサ（MUX）
６６・・・デジタル－アナログ変換器（ＤＡＣ）
８８・・・アナログ・インタフェース制御装置（ＡＩＣ）

９０・・・オーディオ制御装置
９４・・・アダプタ・カード
９６・・・マルチメディア装置（MMD）
９８・・・オペレーティング・システム
１００・・・アプリケーション・プログラム
１０２・・・マルチメディア・アプリケーション・プログラム（MMAP）
１０６・・・メディア・デバイス・マネージャ（MDM）
１１０・・・ＭＩＤＩメディア・ドライバ
１１６・・・動的リンク・ライブラリ（ＤＬＬ）
１３３・・・ストリーム・マネージャ・ヘルパ

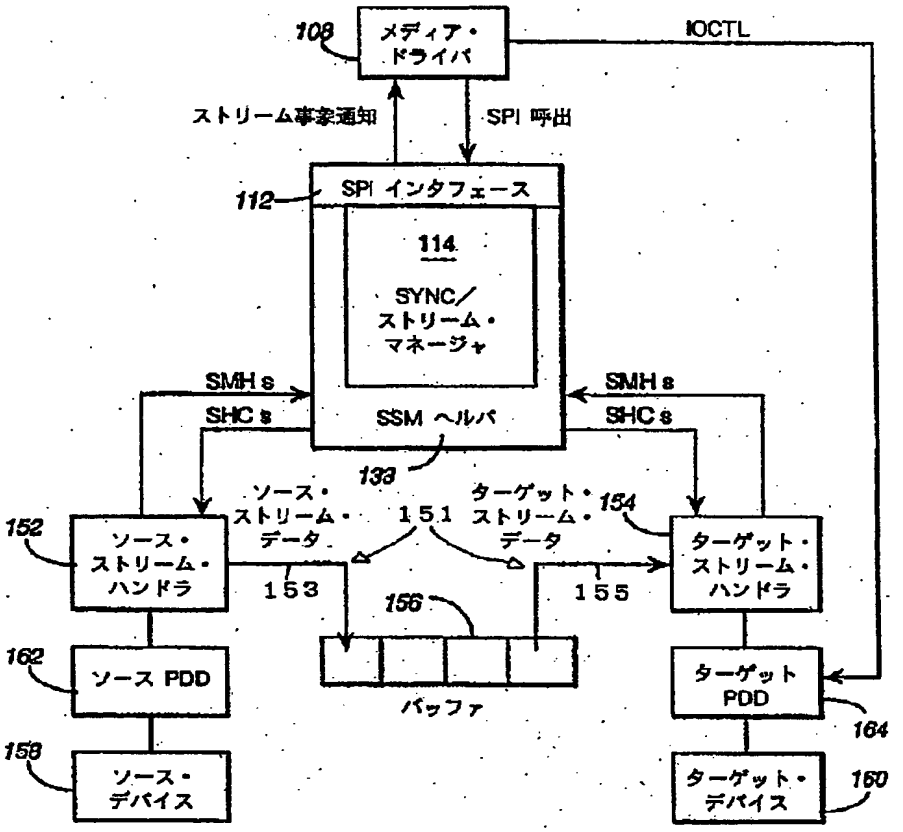
【図１】



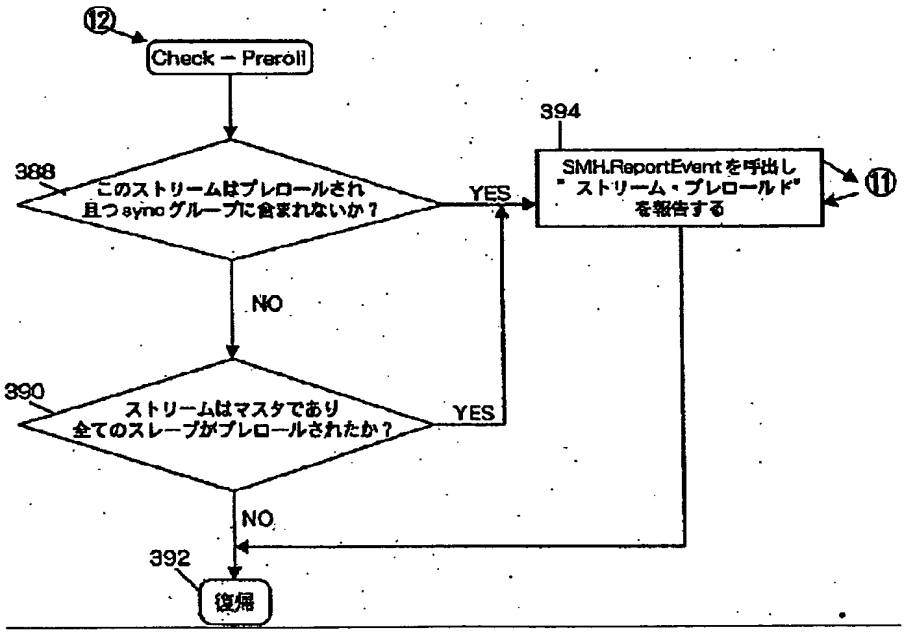
【図２】


102
MMAP
MCI
98
OS
108
メディア・デバイス・マネージャ
116
MIDI マッパー・ストリーム・ハンドラ DLL
118
CD/DA ストリーム・ハンドラ DLL
120
メモリ・ストリーム・ハンドラ DLL
122
スプリット・ストリーム・ハンドラ DLL
CD－ROM/XA ストリーム・ハンドラ DLL
124
オーディオ・メディア・ドライバ
106
MIDI メディア・ドライバ
110
126
ファイル・システム・ストリーム・ハンドラ DLL
128
MMIO マネージャ DLL
130
ファイル・システム
104
112
SPI インタフェース
114
SYNC/ストリーム・マネージャ
SSM ヘルパ
133
リング 3
リング 0
113
146
IDC インタフェース
134
オーディオ・ストリーム・ハンドラ PDD
136
ストリーム・ハンドラ/PDD インタフェース
138
CD－ROM/XA PDD
140
DSP PDD
142
ACPA PDD
144
OEM オーディオ PDD

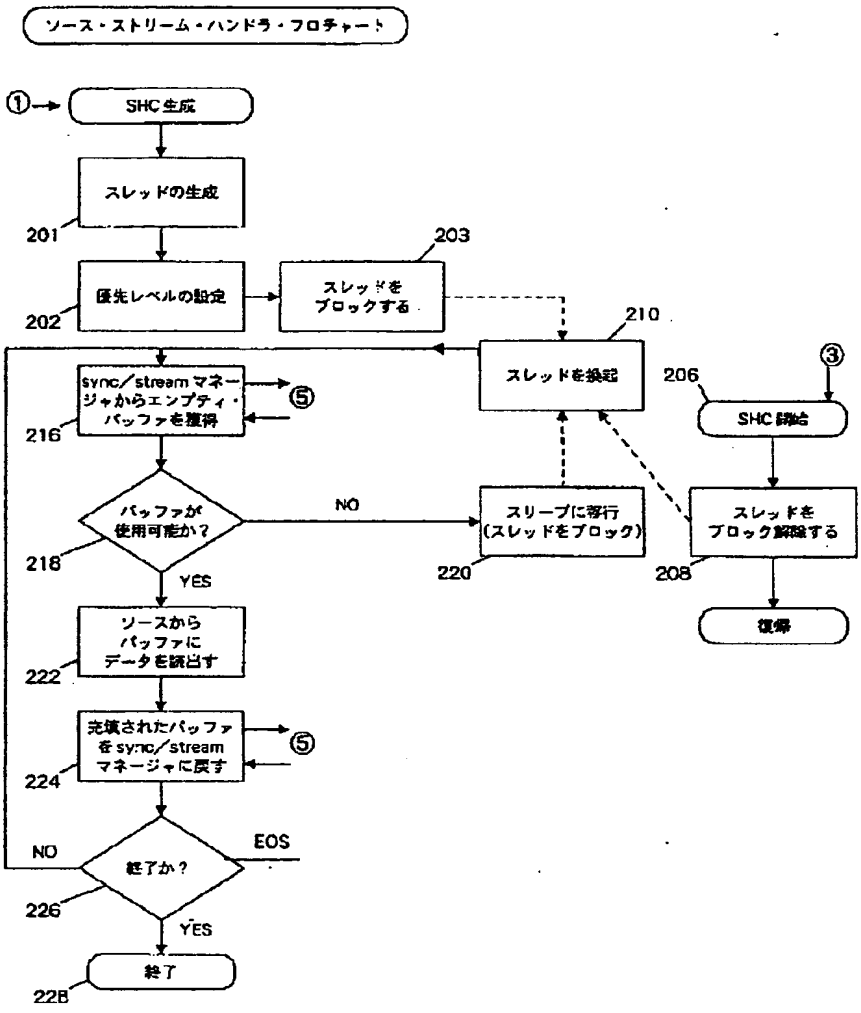
【図３】


108
メディア・ドライバ
IOCTL
ストリーム事象通知
SPI 呼出
112
SPI インタフェース
114
SYNC／ストリーム・マネージャ
SMH s
SHC s
SSM ヘルパ
SMH s
SHC s
133
152
ソース・ストリーム・ハンドラ
ソース・ストリーム・データ
151
ターゲット・ストリーム・データ
154
ターゲット・ストリーム・ハンドラ
153
155
156
バッファ
162
ソース PDD
ターゲット PDD
164
158
ソース・デバイス
ターゲット・デバイス
160

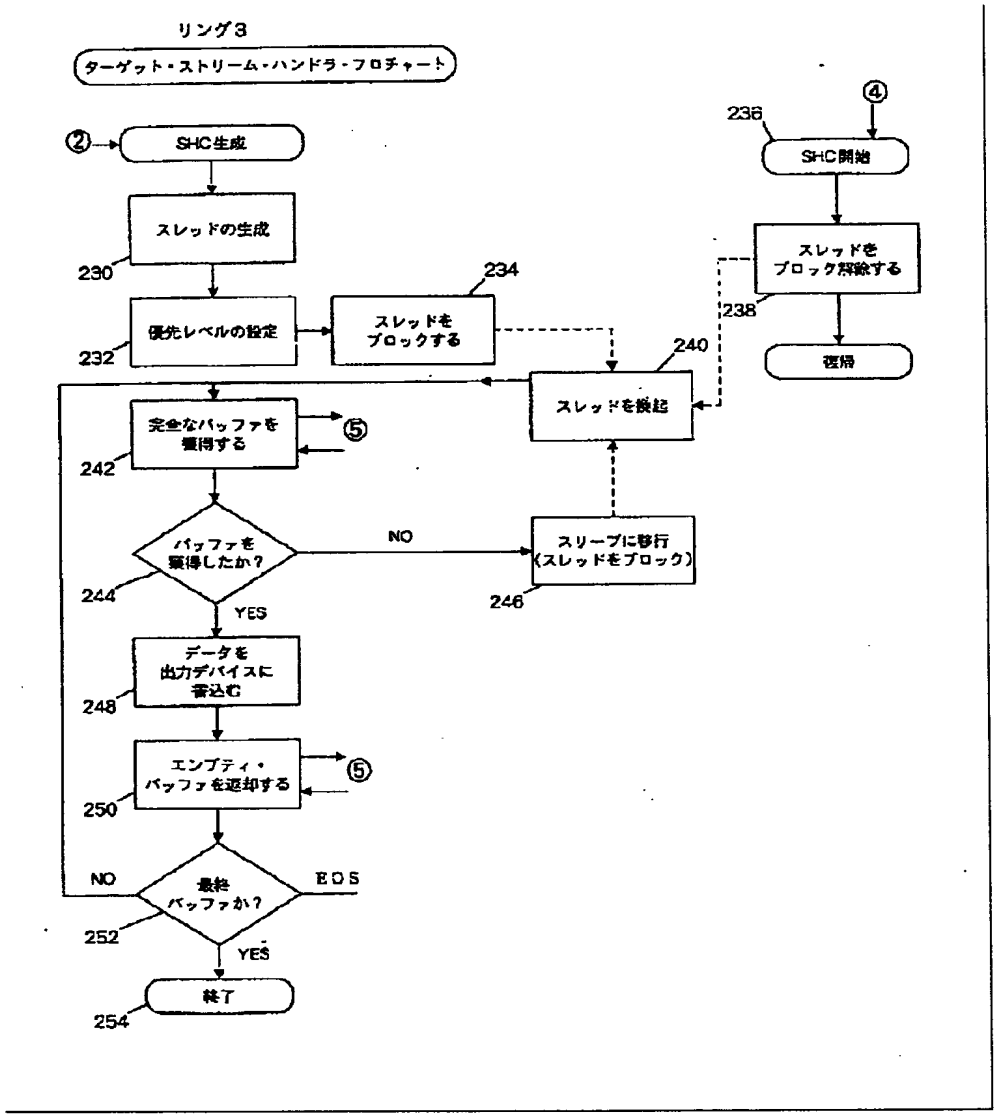
【図１６】


12
Check – Preroll
394
388
このストリームはプレロールされ且つsyncグループに含まれないか？
YES
SMH.ReportEventを呼出し"ストリーム・プレロールド"を報告する
11
NO
390
ストリームはマスタであり全てのスレーブがプレロールされたか？
YES
NO
392
復帰

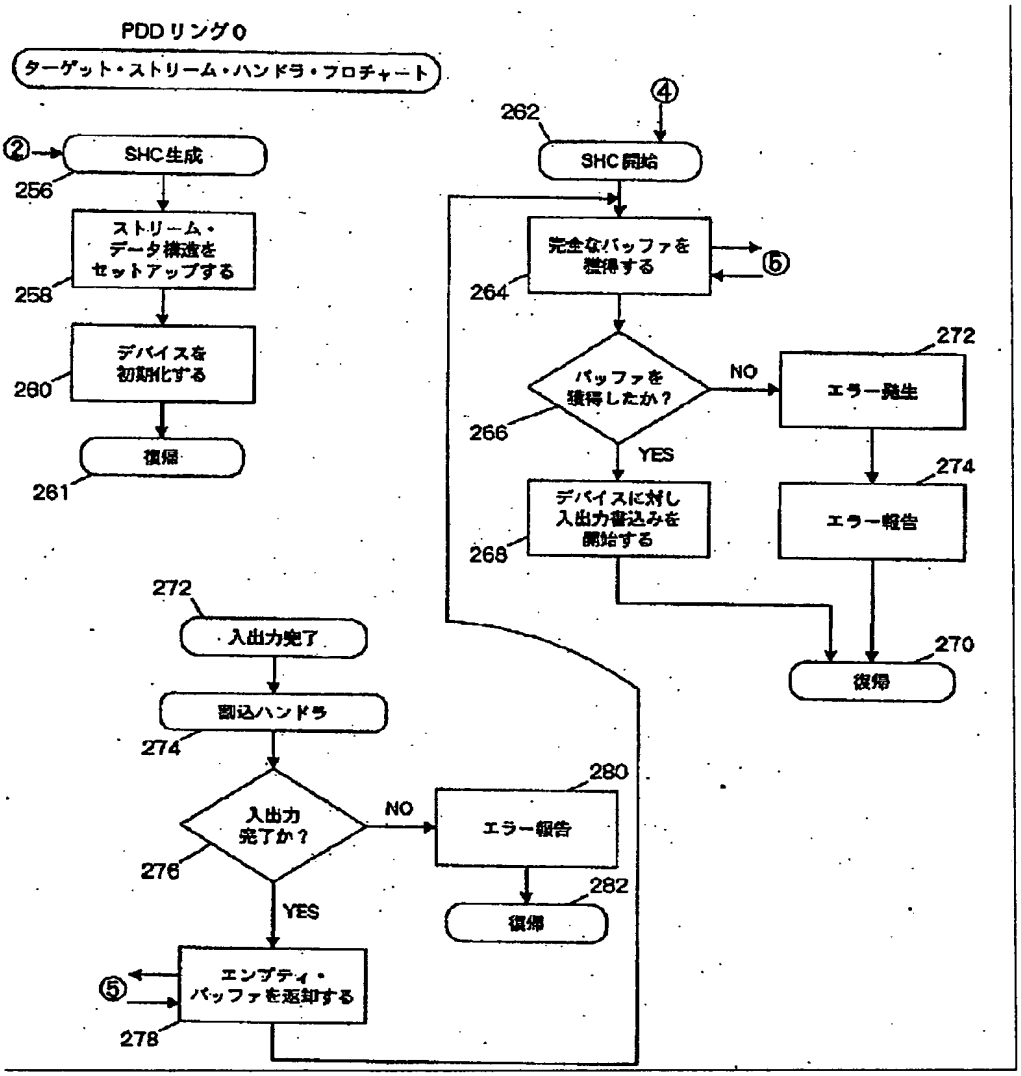
【図４】


ソース・ストリーム・ハンドラ・フローチャート
①
SHC生成
スレッドの生成
201
優先レベルの設定
202
スレッドをブロックする
203
スレッドを換起
210
③
206
SHC開始
sync/streamマネージャからエンプティ・バッファを獲得
⑤
216
バッファが使用可能か？
218
NO
スリープに移行（スレッドをブロック）
220
スレッドをブロック解除する
208
YES
ソースからバッファにデータを読出す
222
充填されたバッファをsync/streamマネージャに戻す
⑤
224
NO
終了か？
EOS
226
YES
終了
228
復帰

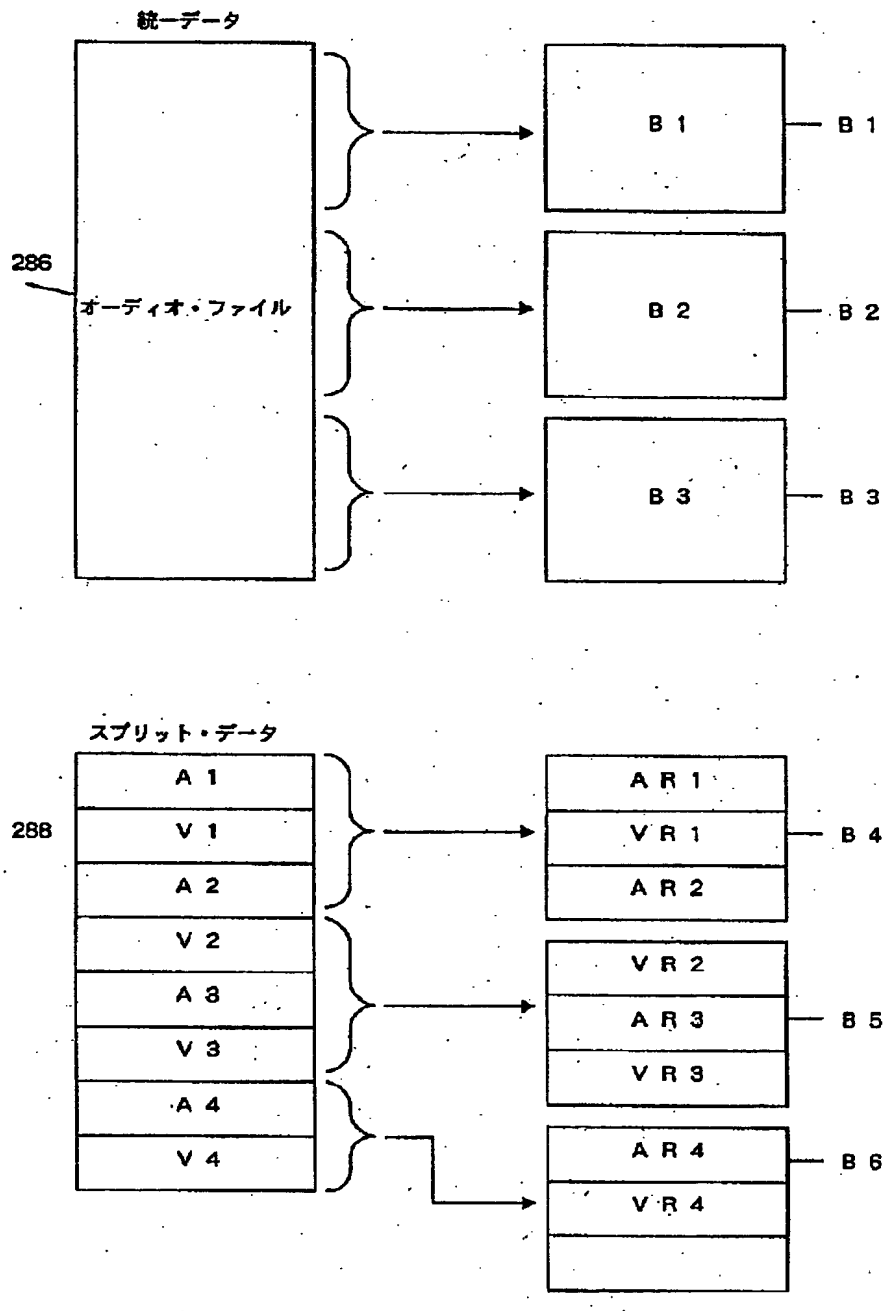
【図５】


リング3
ターゲット・ストリーム・ハンドラ・プロチャート
②
SHC生成
スレッドの生成
230
優先レベルの設定
232
スレッドを
ブロックする
234
スレッドを起起
240
④
236
SHC開始
スレッドを
ブロック解除する
238
復帰
完全なバッファを
獲得する
242
⑤
バッファを
獲得したか?
244
NO
スリープに移行
(スレッドをブロック)
246
YES
データを
出力デバイスに
書込む
248
エンプティ・
バッファを返却する
250
⑤
NO
最終
バッファか?
252
EOS
YES
終了
254

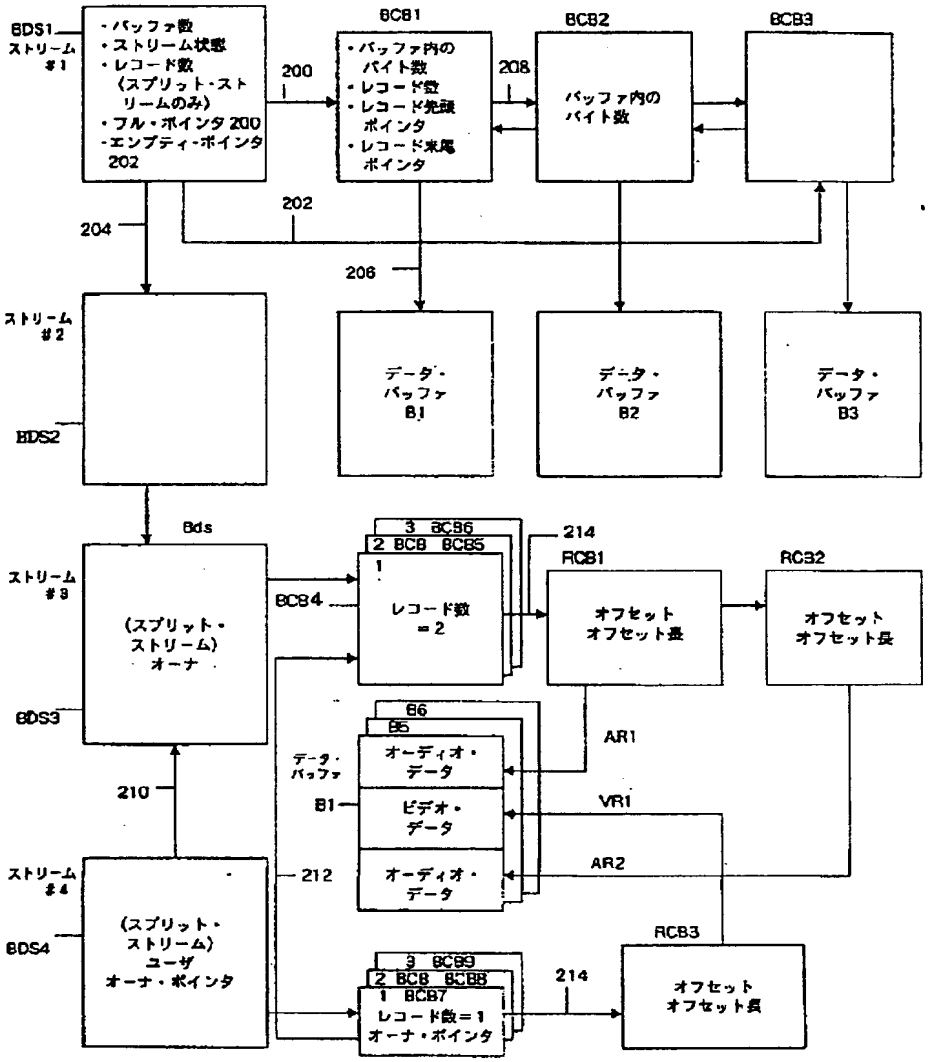
【図６】


PDDリング０
ターゲット・ストリーム・ハンドラ・プロチャート
②
SHC生成
256
ストリーム・データ構造をセットアップする
258
デバイスを初期化する
260
復帰
261
④
262
SHC開始
完全なバッファを獲得する
⑤
264
バッファを獲得したか？
NO
272
エラー発生
266
YES
デバイスに対し入出力書込みを開始する
268
エラー報告
274
復帰
270
272
入出力完了
割込ハンドラ
274
入出力完了か？
NO
280
エラー報告
276
282
復帰
YES
⑤
エンプティ・バッファを返却する
278

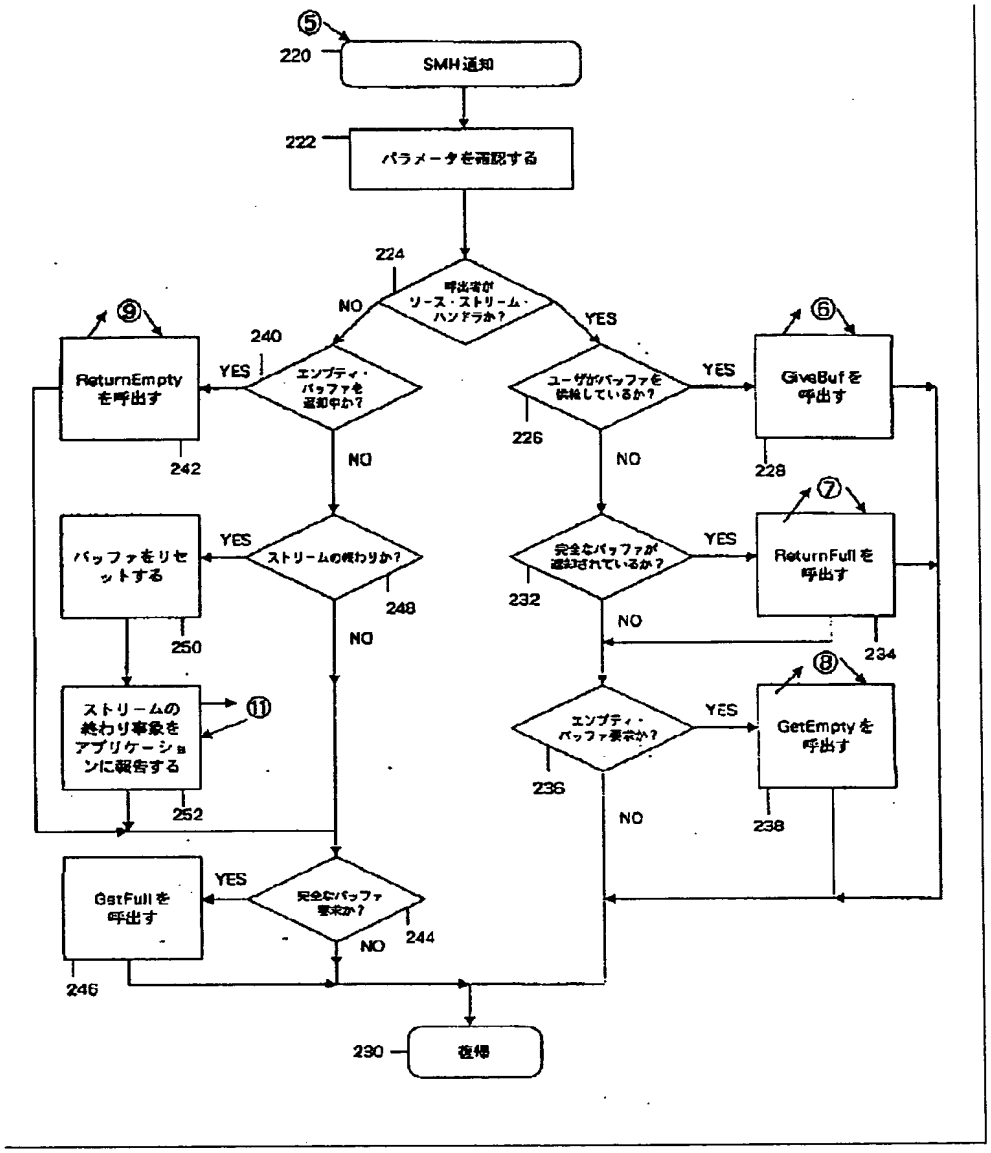
【図7】


統一データ
286
オーディオ・ファイル
B 1
B 2
B 3
B 1
B 2
B 3
スプリット・データ
288
A 1
V 1
A 2
V 2
A 3
V 3
A 4
V 4
A R 1
V R 1
A R 2
V R 2
A R 3
V R 3
A R 4
V R 4
B 4
B 5
B 6

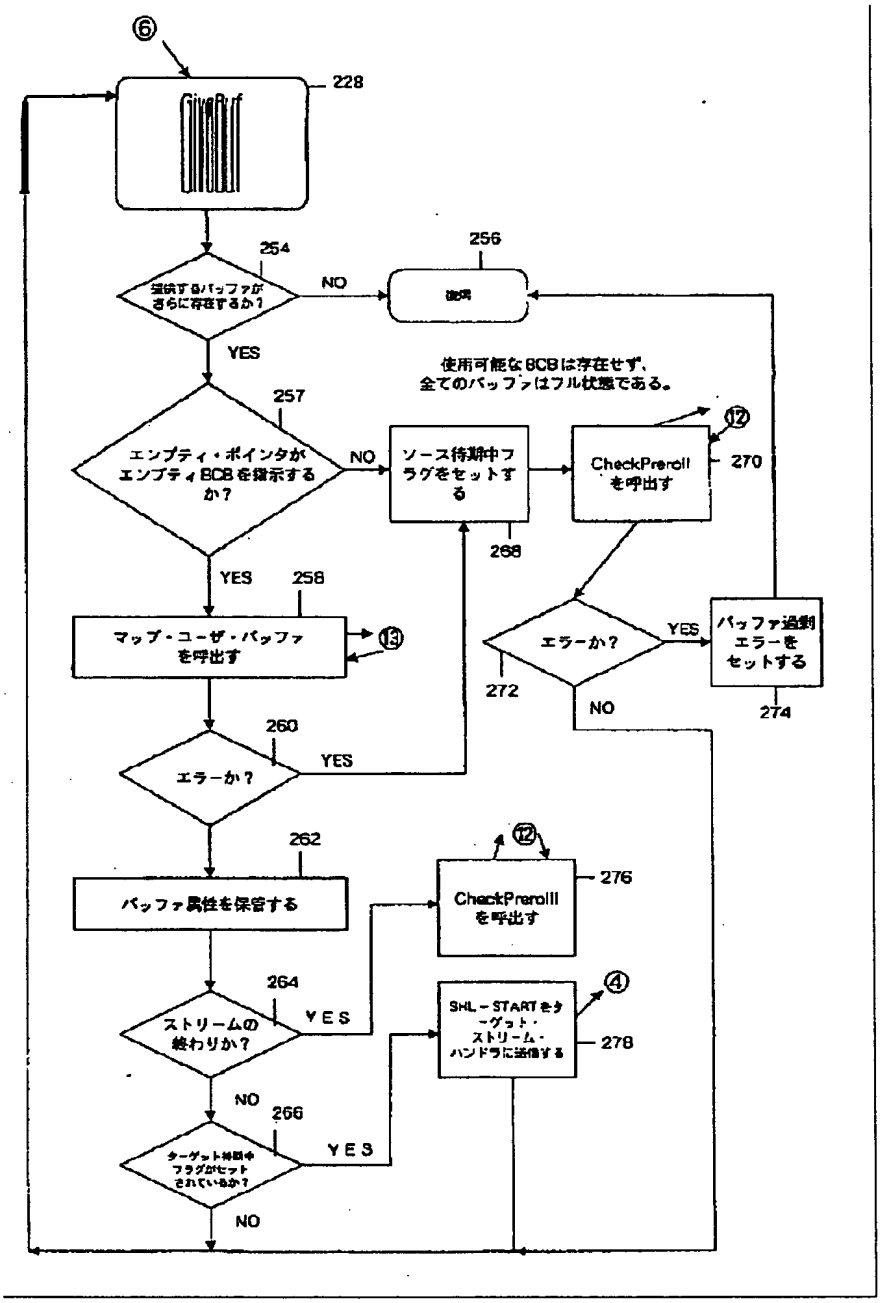
【図８】


BDS1
ストリーム #1
・バッファ数
・ストリーム状態
・レコード数
（スプリット・ストリームのみ）
・フル・ポインタ 200
・エンプティ・ポインタ 202
200
BCB1
・バッファ内のバイト数
・レコード数
・レコード先頭ポインタ
・レコード末尾ポインタ
208
BCB2
バッファ内のバイト数
BCB3
202
204
206
ストリーム #2
BDS2
データ・バッファ B1
データ・バッファ B2
データ・バッファ B3
Bds
ストリーム #3
（スプリット・ストリーム）オーナ
BDS3
3 BCB6
2 BCB BCB5
1
BCB4
レコード数＝2
214
RCB1
オフセット
オフセット長
RCB2
オフセット
オフセット長
B6
B5
データ・バッファ
B1
オーディオ・データ
ビデオ・データ
オーディオ・データ
AR1
VR1
AR2
210
212
ストリーム #4
BDS4
（スプリット・ストリーム）ユーザ オーナ・ポインタ
3 BCB9
2 BCB BCB8
1 BCB7
レコード数＝1
オーナ・ポインタ
214
RCB3
オフセット
オフセット長

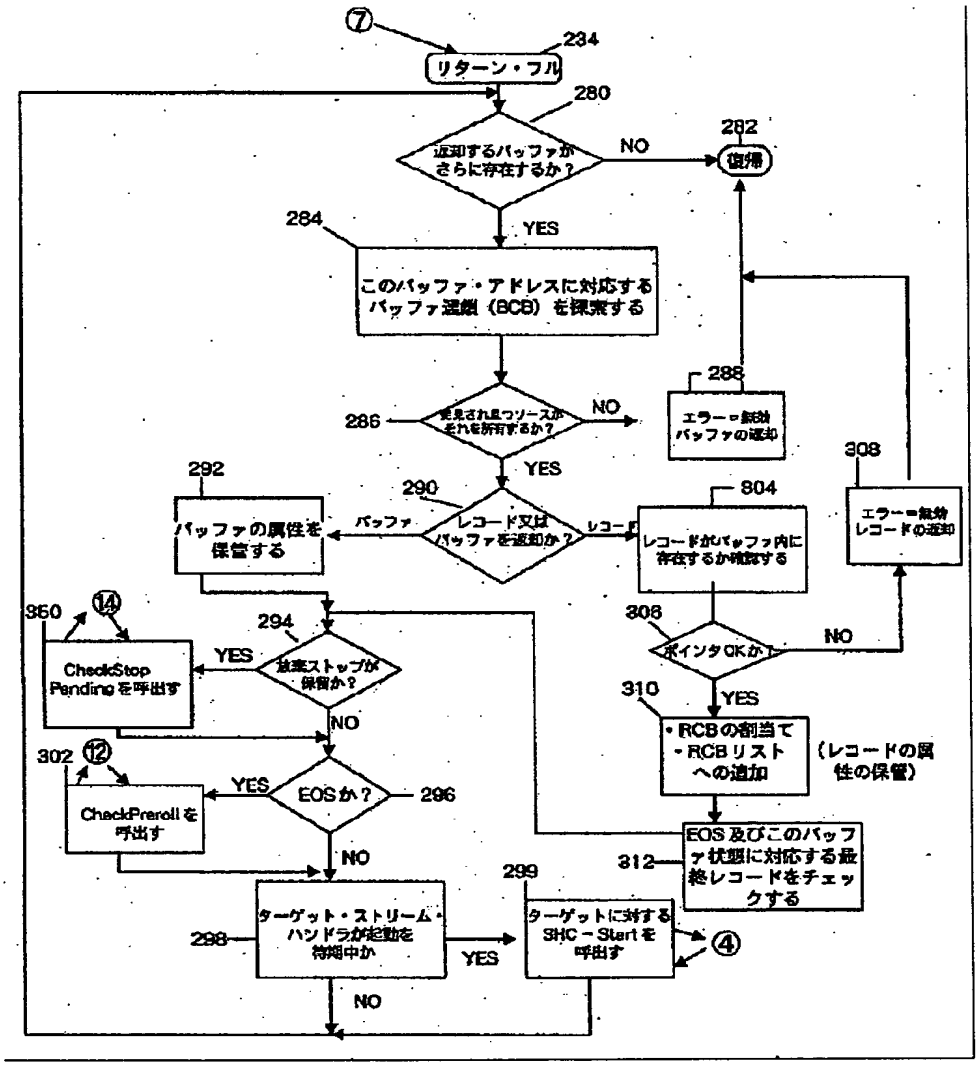
【図９】


⑤
220
SMH通知
222
パラメータを確認する
224
呼出者がソース・ストリーム・ハンドラか？
NO
YES
⑨
ReturnEmpty を呼出す
242
240
エンプティ・バッファを返却中か？
YES
NO
⑥
GiveBuf を呼出す
228
ユーザがバッファを保持しているか？
226
YES
NO
⑦
ReturnFull を呼出す
234
完全なバッファが返却されているか？
232
YES
NO
⑧
GetEmpty を呼出す
238
エンプティ・バッファ要求か？
236
YES
NO
バッファをリセットする
250
ストリームの終わりか？
248
YES
NO
ストリームの終わり事象をアプリケーションに報告する
252
⑪
GstFull を呼出す
246
完全なバッファ要求か？
244
YES
NO
230
復帰

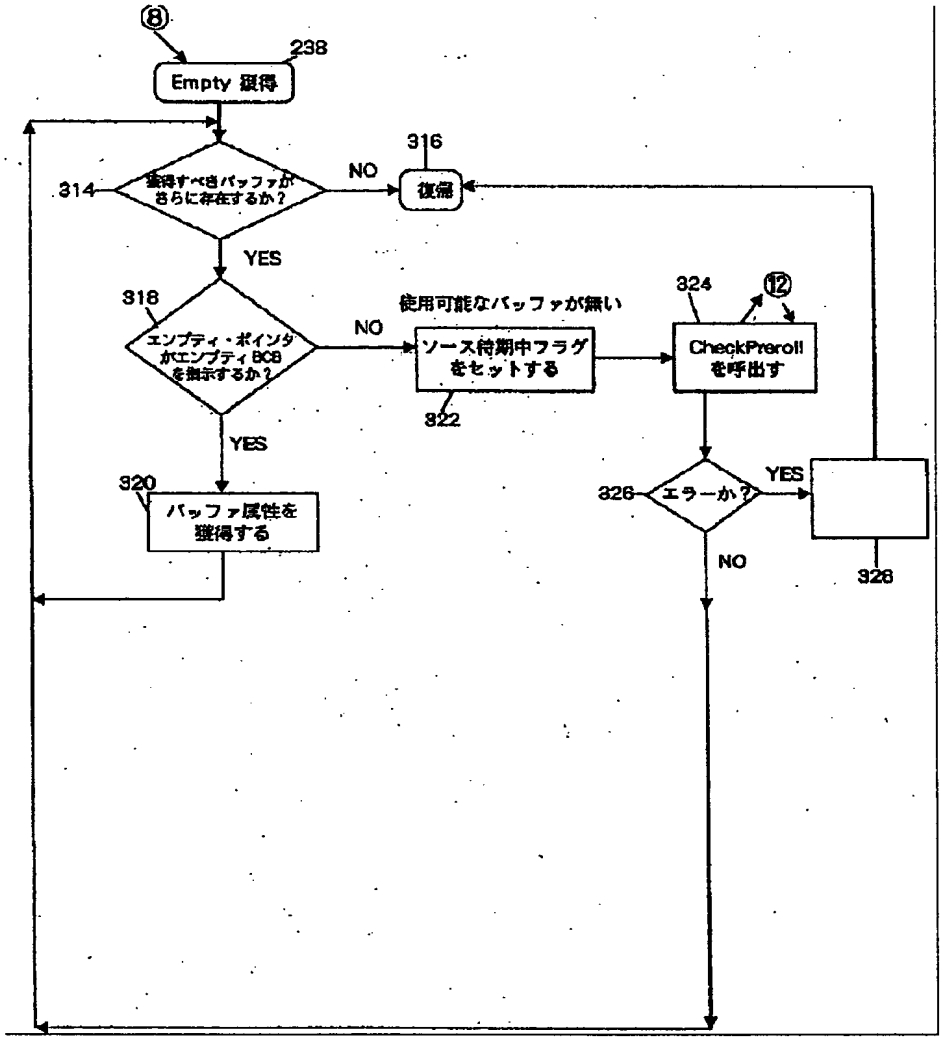
【図１０】


⑥
228
254
NO
256
YES
使用可能なBCBは存在せず、
全てのバッファはフル状態である。
257
NO
⑫
270
YES
258
⑬
YES
272
NO
274
260
YES
262
⑫
276
264
YES
④
278
NO
266
YES
NO

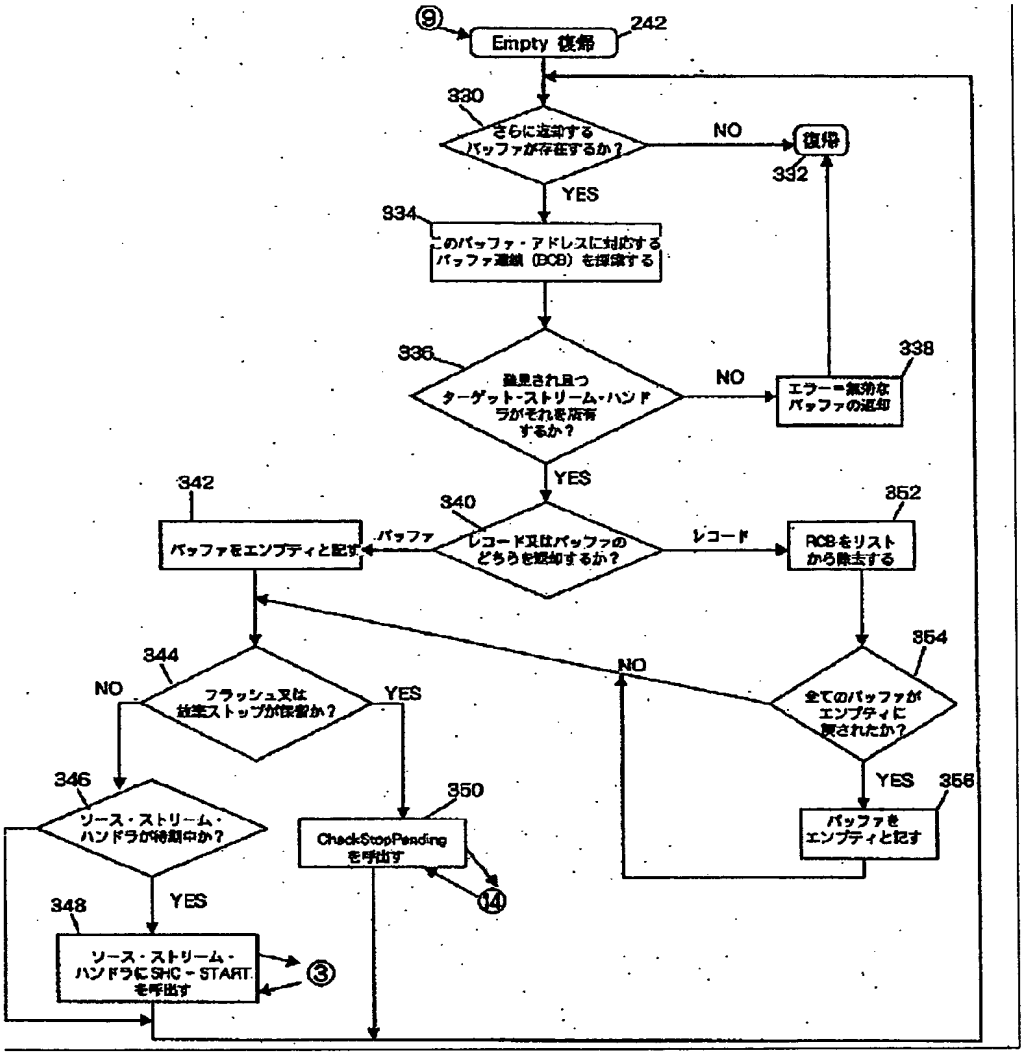
【図１１】


⑦
234
リターン・フル
280
返却するバッファがさらに存在するか？
NO
282
復帰
YES
284
このバッファ・アドレスに対応するバッファ記述子（BCB）を探索する
286
要求されたリソースがそれを所有するか？
NO
288
エラー＝無効バッファの返却
YES
290
レコード又はバッファを返却か？
バッファ
レコード
292
バッファの属性を保管する
804
レコードがバッファ内に存在するか確認する
308
エラー＝無効レコードの返却
306
ポインタOKか？
NO
YES
310
・RCBの割当て
・RCBリストへの追加
（レコードの属性の保管）
312
EOS及びこのバッファ状態に対応する最終レコードをチェックする
350
⑭
CheckStop Pendingを呼出す
294
YES
放棄ストップが保留か？
NO
302
⑫
CheckPrerollを呼出す
YES
EOSか？
296
NO
298
ターゲット・ストリーム・ハンドラが起動を待機中か
YES
299
ターゲットに対するSHC－Startを呼出す
④
NO

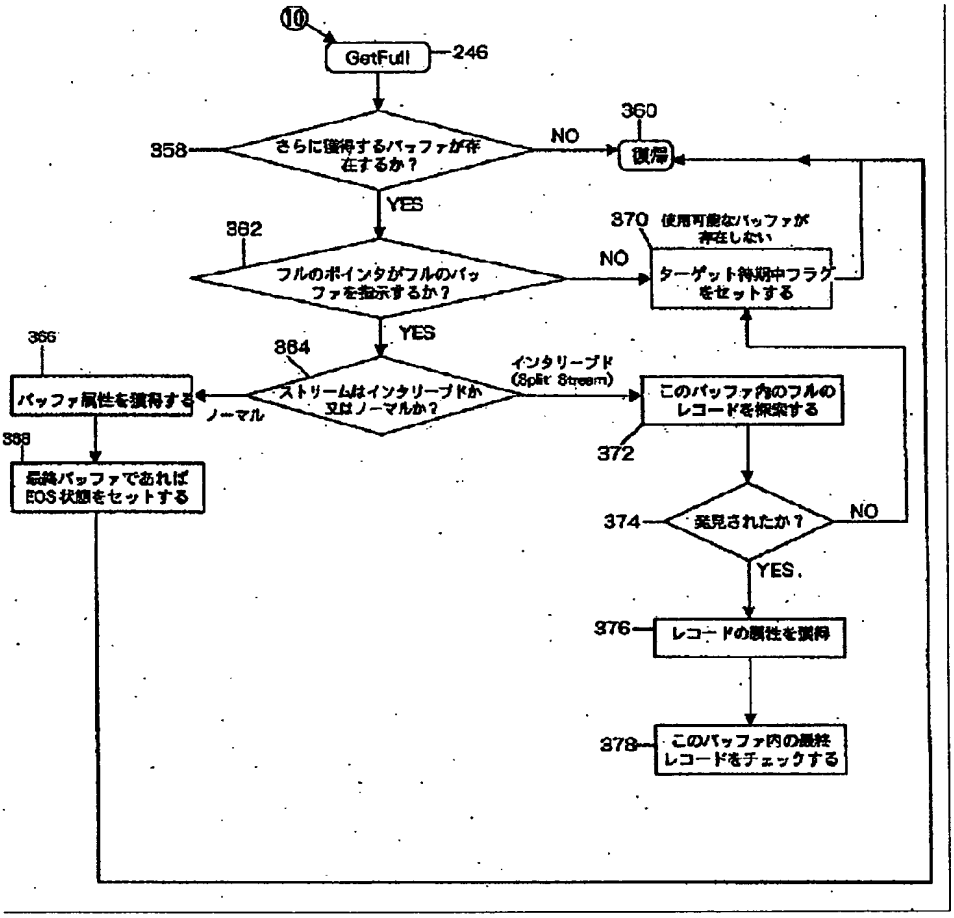
【図１２】


⑧
238
Empty 獲得
316
314
獲得すべきバッファがさらに存在するか？
NO
復帰
YES
318
エンプティ・ポインタがエンプティBCBを指示するか？
使用可能なバッファが無い
NO
ソース待期中フラグをセットする
322
324
⑫
CheckPreroll を呼出す
YES
320
バッファ属性を獲得する
326
エラーか？
YES
NO
328

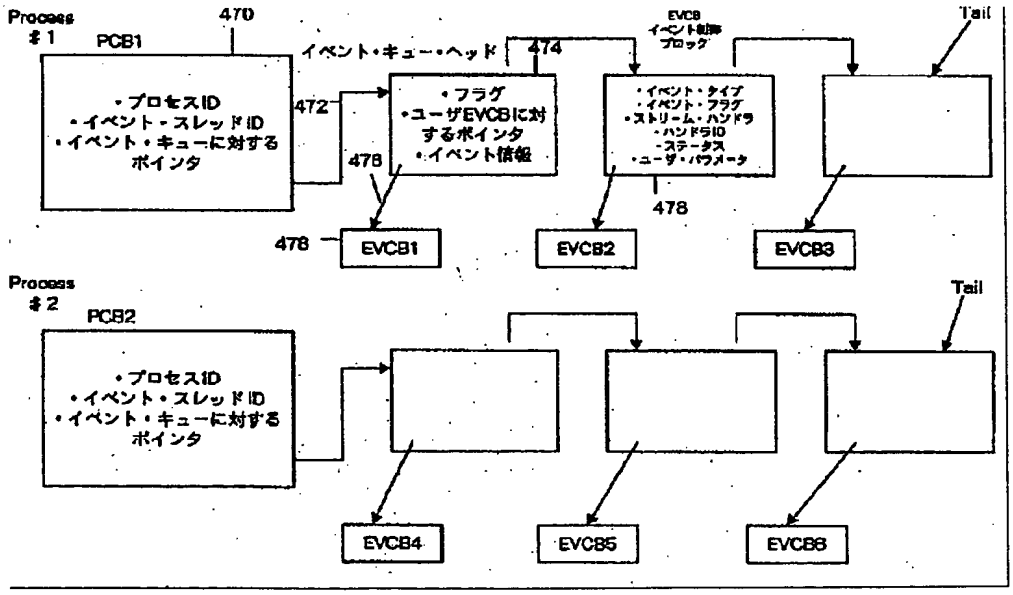
【図１３】


⑨
Empty 復帰
242
330
さらに返却するバッファが存在するか？
NO
復帰
332
YES
334
このバッファ・アドレスに対応するバッファ連鎖（BCB）を探索する
336
識別され且つターゲット・ストリーム・ハンドラがそれを所有するか？
NO
338
エラー無効をバッファの返却
YES
342
340
バッファをエンプティと記す
バッファ
レコード又はバッファのどちらを返却するか？
レコード
352
RCBをリストから除去する
344
NO
フラッシュ又は放棄ストップが保留か？
YES
NO
354
全てのバッファがエンプティに戻されたか？
346
ソース・ストリーム・ハンドラが待期中か？
350
CheckStopPending を呼出す
YES
356
バッファをエンプティと記す
348
YES
ソース・ストリーム・ハンドラにSHC－START を呼出す
③
⑭

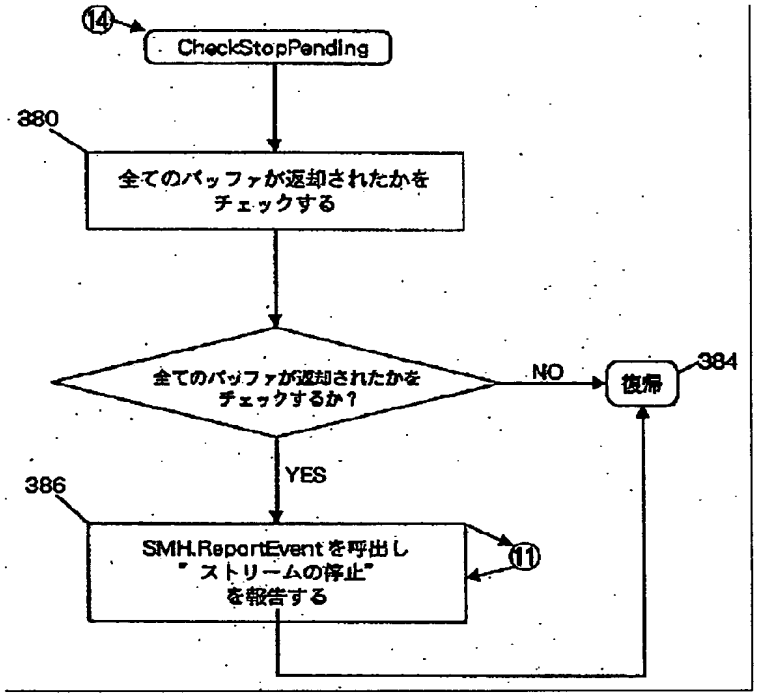
【図１４】


10
GetFull
246
さらに獲得するバッファが存在するか？
358
NO
360
復帰
YES
362
フルのポインタがフルのバッファを指示するか？
NO
370
使用可能なバッファが存在しない
ターゲット待期中フラグをセットする
YES
364
ストリームはインタリーブドか又はノーマルか？
インタリーブド (Split Stream)
ノーマル
366
バッファ属性を獲得する
368
最終バッファであればEOS状態をセットする
このバッファ内のフルのレコードを探索する
372
374
発見されたか？
NO
YES
376
レコードの属性を獲得
378
このバッファ内の最終レコードをチェックする

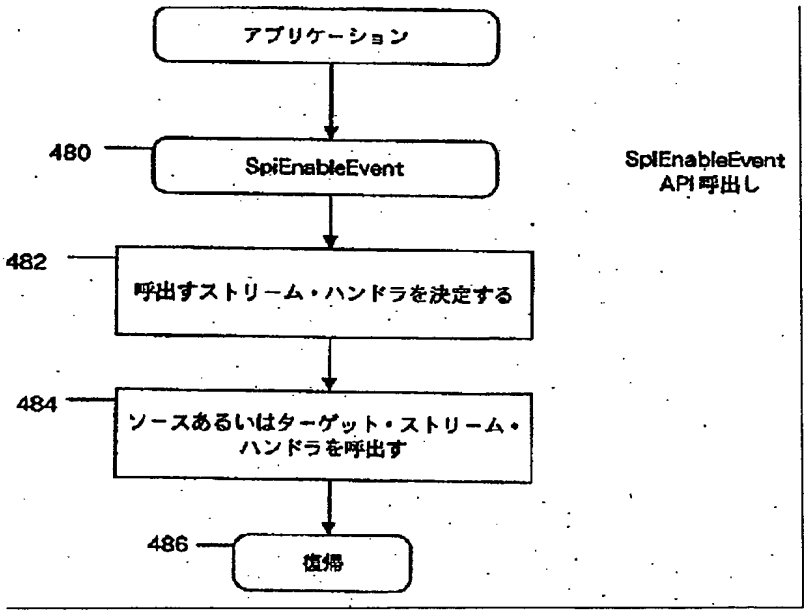
【図２２】


Process #1
PCB1
470
・プロセスID
・イベント・スレッドID
・イベント・キューに対するポインタ
イベント・キュー・ヘッド
474
472
・フラグ
・ユーザEVCBに対するポインタ
・イベント情報
478
EVCB
イベント制御ブロック
・イベント・タイプ
・イベント・フラグ
・ストリーム・ハンドラ
・ハンドラID
・ステータス
・ユーザ・パラメータ
Tail
EVCB1
EVCB2
EVCB3
Process #2
PCB2
・プロセスID
・イベント・スレッドID
・イベント・キューに対するポインタ
EVCB4
EVCB5
EVCB6

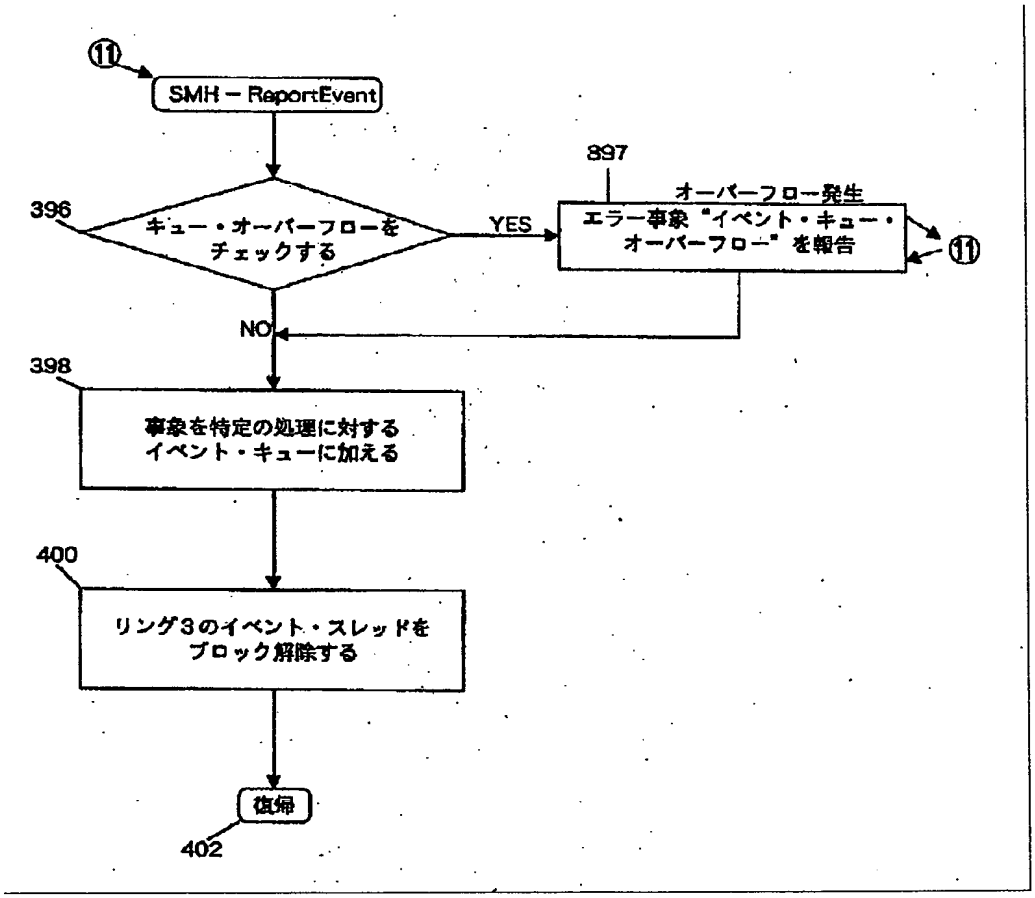
【図１５】


⑭
CheckStopPending
380
全てのバッファが返却されたかを
チェックする
全てのバッファが返却されたかを
チェックするか？
NO
復帰
384
YES
386
SMH.ReportEventを呼出し
" ストリームの停止"
を報告する
⑪

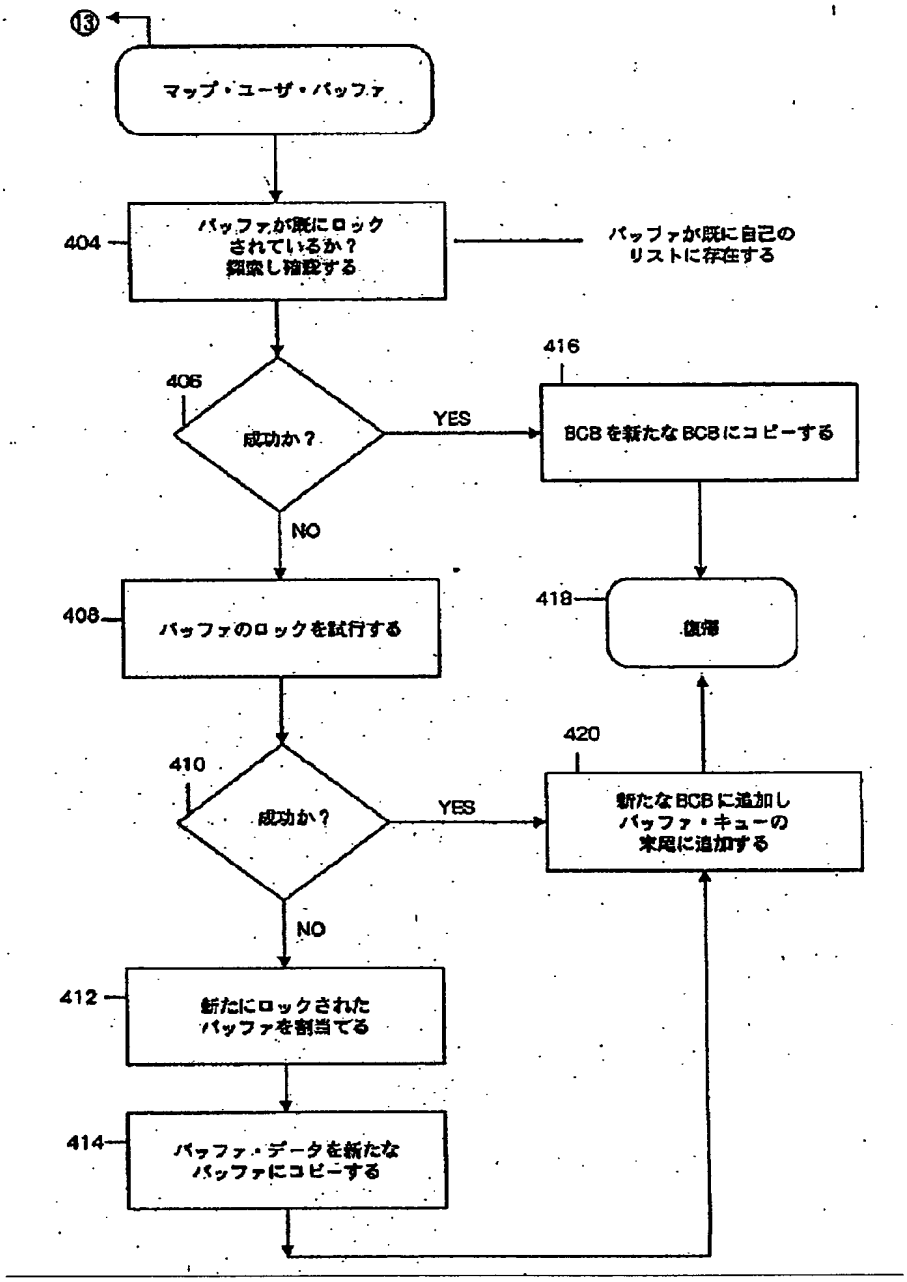
【図２３】


アプリケーション
480
SpiEnableEvent
SpiEnableEvent
API呼出し
482
呼出すストリーム・ハンドラを決定する
484
ソースあるいはターゲット・ストリーム・
ハンドラを呼出す
486
復帰

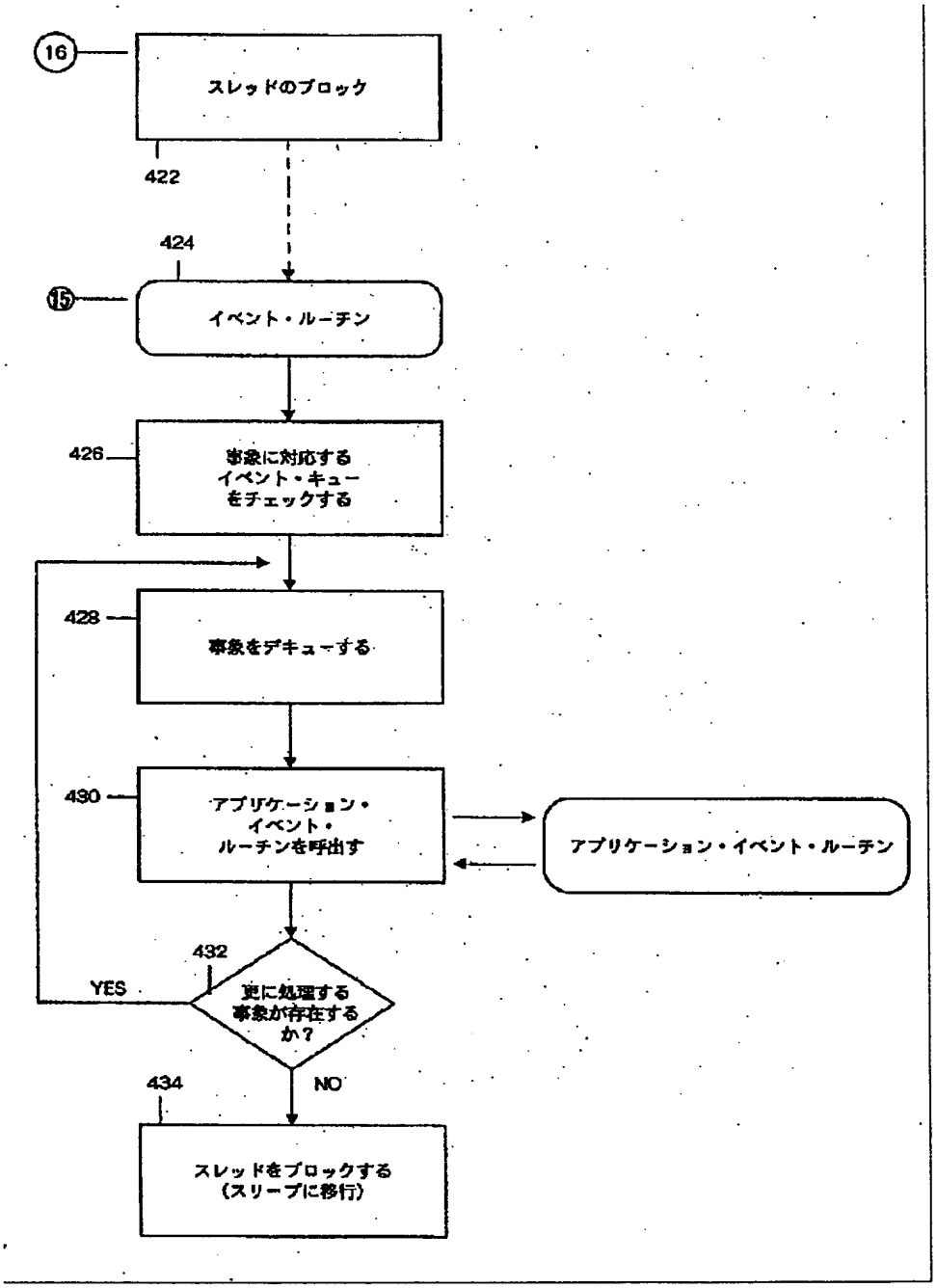
【図１７】


⑪
SMH－ReportEvent
396
キュー・オーバーフローを
チェックする
YES
897
オーバーフロー発生
エラー事象“イベント・キュー・
オーバーフロー”を報告
⑪
NO
398
事象を特定の処理に対する
イベント・キューに加える
400
リング３のイベント・スレッドを
ブロック解除する
復帰
402

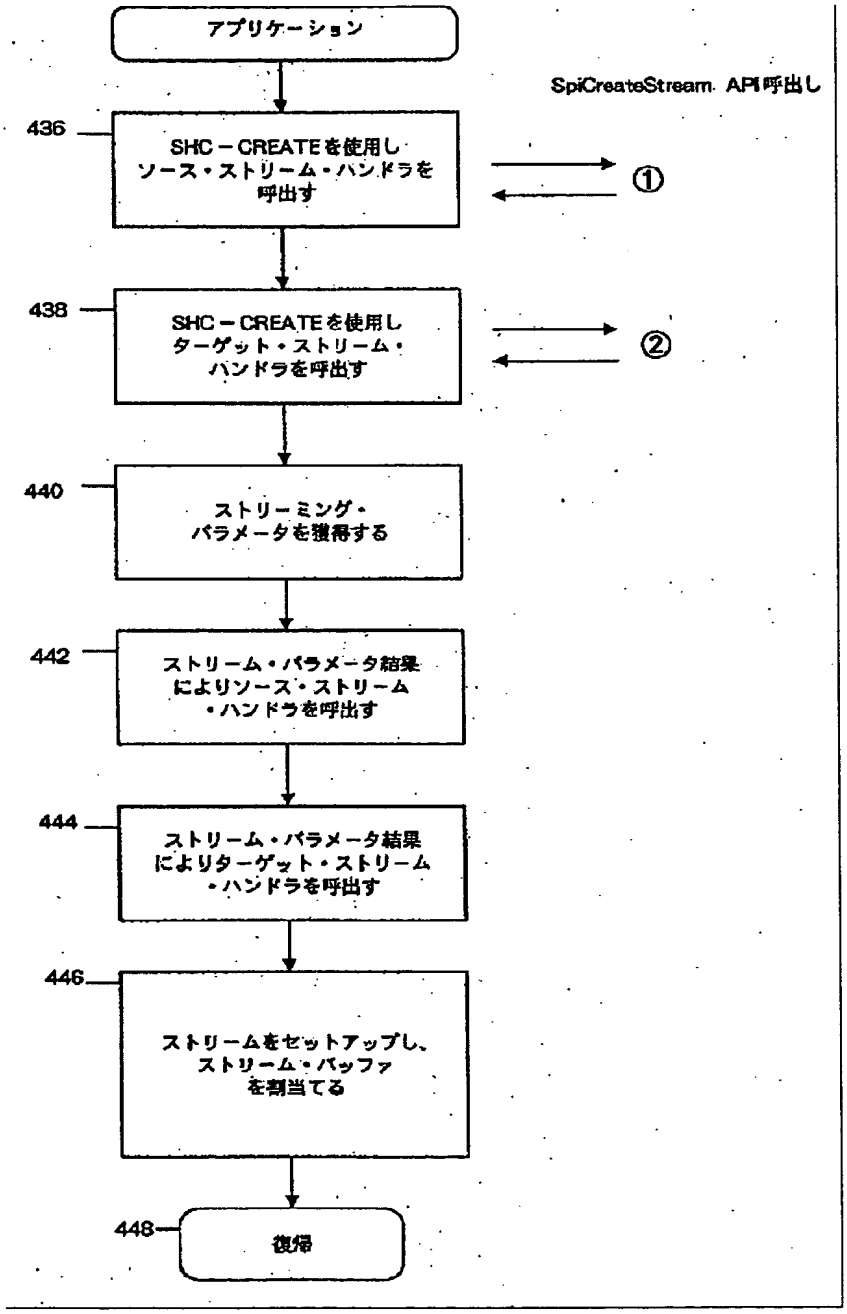
【図１８】


⑬
マップ・ユーザ・バッファ
404
バッファが既にロック
されているか？
探索し格納する
バッファが既に自己の
リストに存在する
406
成功か？
YES
416
BCBを新たなBCBにコピーする
NO
408
バッファのロックを試行する
418
復帰
410
成功か？
YES
420
新たなBCBに追加し
バッファ・キューの
末尾に追加する
NO
412
新たにロックされた
バッファを割当てる
414
バッファ・データを新たな
バッファにコピーする

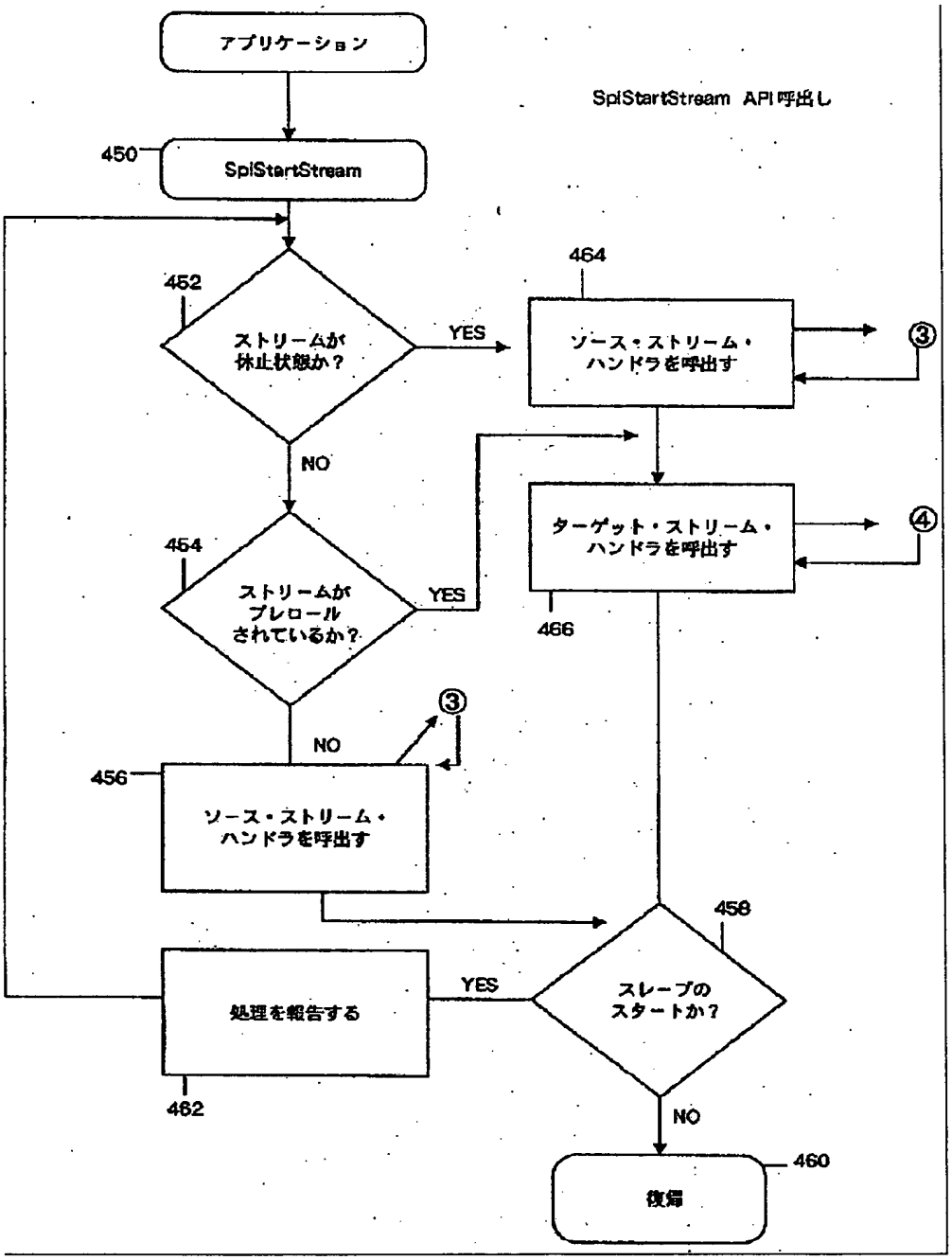
【図１９】


16
スレッドのブロック
422
424
15
イベント・ルーチン
426
事象に対応する
イベント・キュー
をチェックする
428
事象をデキューする
430
アプリケーション・
イベント・
ルーチンを呼出す
アプリケーション・イベント・ルーチン
432
YES
更に処理する
事象が存在する
か？
NO
434
スレッドをブロックする
（スリープに移行）

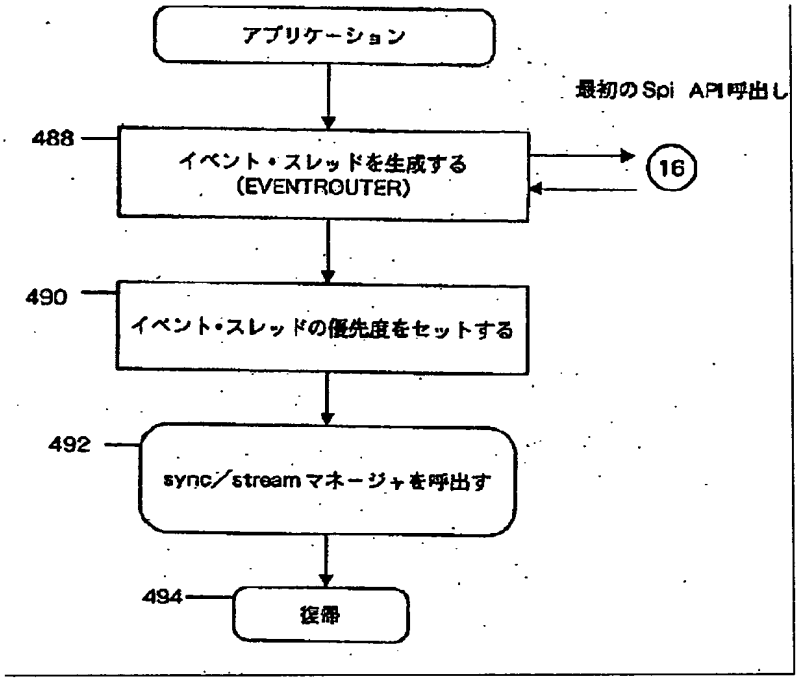
【図２０】


アプリケーション
SpiCreateStream API呼出し
436
SHC－CREATEを使用し
ソース・ストリーム・ハンドラを
呼出す
①
438
SHC－CREATEを使用し
ターゲット・ストリーム・
ハンドラを呼出す
②
440
ストリーミング・
パラメータを獲得する
442
ストリーム・パラメータ結果
によりソース・ストリーム
・ハンドラを呼出す
444
ストリーム・パラメータ結果
によりターゲット・ストリーム
・ハンドラを呼出す
446
ストリームをセットアップし、
ストリーム・バッファ
を割当てる
448
復帰

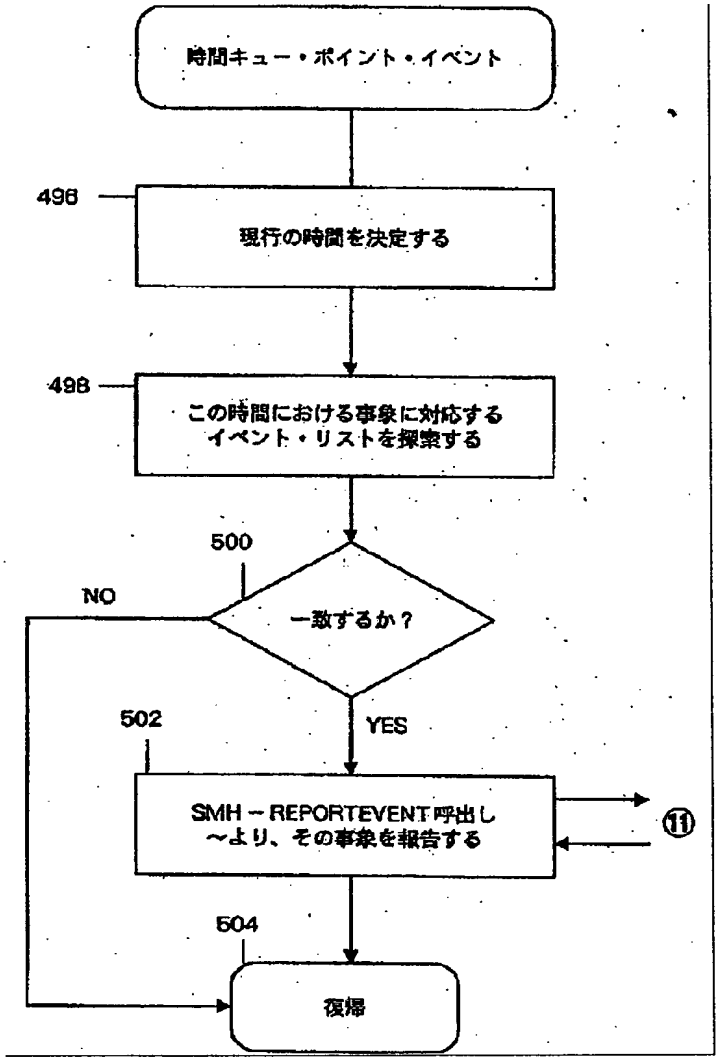
【図２１】


アプリケーション
SplStartStream API呼出し
450
SplStartStream
452
ストリームが
休止状態か？
YES
464
ソース・ストリーム・
ハンドラを呼出す
③
NO
ターゲット・ストリーム・
ハンドラを呼出す
④
454
ストリームが
プレロール
されているか？
YES
466
③
NO
456
ソース・ストリーム・
ハンドラを呼出す
458
処理を報告する
YES
スレーブの
スタートか？
462
NO
460
復帰

【図２４】


アプリケーション
最初のSpi API呼出し
488
イベント・スレッドを生成する
(EVENTROUTER)
16
490
イベント・スレッドの優先度をセットする
492
sync／streamマネージャを呼出す
494
復帰

【図２５】



フロントページの続き

(72)発明者　マイケル・ジェイ・コバル
アメリカ合衆国33496、フロリダ州ボカ・ラトン、アファームド・レーン 9080
(72)発明者　ウイリアム・ダブリュ・ロートン
アメリカ合衆国33433、フロリダ州ボカ・ラトン、バターフィールド・レーン 8290
(72)発明者　マーティン・ジェイ・ポーラット、ジュニア
アメリカ合衆国33444、フロリダ州デルレイ・ビーチ、ゼダー・アベニュー 2409
(72)発明者　ジョン・ジー・タイラー
アメリカ合衆国33437、フロリダ州ボイントン・ビーチ、ランチパー・ストリート 10686

(72)発明者　スコット・エル・ウインタース
アメリカ合衆国33324、フロリダ州プランテーション、ノースウエスト・シックス・ストリート　10309

(72)発明者　ゲリー・ジー・オールラン
アメリカ合衆国33431、フロリダ州ボカ・ラトン、、オーチャード・リジ・レーン　117

(56)参考文献　特開　昭63－186336（ＪＰ，Ａ）
実開　平２－145449（ＪＰ，Ｕ）